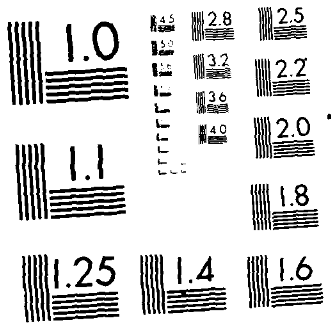
AD-A173 886 PROCEEDINGS OF THE INTERNATIONAL MEETING ON ADVANCES IN LEARNING HELD IN..(U) CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE ORSAY (FRANCE) L.. D J GILMORE ET AL. 1/3

UNCLASSIFIED 01 AUG 86 RR-290 F/G 5/10 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

4

L.R.I

RAPPORTS DE RECHERCHE

PROCEEDINGS OF THE

INTERNATIONAL MEETING

ON ADVANCES IN LEARNING

I M A L ' 8 6

Les Arcs 28th - August 1986

UNIVERSITE PARIS-SUD

Centre d'Orsay

LABORATOIRE DE RECHERCHE EN INFORMATIQUE

Bât. 490

91405 ORSAY (France)

86 1 [illegible] 0

PROCEEDINGS OF THE
INTERNATIONAL MEETING
ON ADVANCES IN LEARNING

IMAL ~ 86

Les Arc, July 28th - August 1st 1986

Unité Associée au CNRS UA 410, AL KHOWARIZMI

Rapport de Recherche N° 290

[illegible]

A-1

PROCEEDINGS OF IMAL - 86

INTERNATIONAL MEETING ON ADVANCES IN LEARNING

Les Arcs (France) July 28th - August 1st

ACKNOWLEDGEMENTS

This meeting is organized by "Association Française pour l'Apprentissage Symbolique Automatique". Its initiators and organizers are Y. Kodratoff and R. Michalski.

It has been sponsored by USARG, DIGITAL, and CGE.

Without funding of USARG and DIGITAL, its organization would have been impossible.
Let us thank their real interest in Artificial Intelligence and Machine learning.

PROGRAMME COMMITTEE

The programme committee is made of

the invited speakers :
Yves Kodratoff, Pat Langley, Michael Lebowitz, Ryszard Michalski, Bruce Porter, Tom Mitchell, Roger Shank.

the commentators :
I. Bratko, P. Brazdil, R. Holte, R. Stepp.

Partial Contents:

SUMMARY

The papers of Mitchell, Lebowitz, Michalski, Langley & Nordhausen, Shank were received too late for inclusion in this volume. They will be distributed to participants at the beginning of the meeting.

P 3

CERCLE TECHNICAL REPORT NO. 6

CONCEPT LEARNING: ALTERNATIVE METHODS OF FOCUSSING

BY

DAVID J. GILMORE

Centre for Research on Computers and Learning

University of Lancaster
Lancaster
LA1 4YR

P 4

CONCEPT LEARNING: ALTERNATIVE METHODS OF FOCUSSING

by

David J. Gilmore,

Centre for Research on Computers and Learning,
University of Lancaster,
Lancaster, LA1 4YR.
England.

ABSTRACT

This paper discusses the standard focussing algorithm and discusses its strengths and weaknesses. Two alternative versions of the algorithm (POSNEG and MULTI) are presented, which do not suffer from some of the weaknesses of the standard algorithm, and which can learn some disjunctive concepts. A closer analysis of the standard algorithm and POSNEG reveals that they are closely related to a common underlying algorithm, but that they make different assumptions about how to take advantage of the structure of the description space. These assumptions are termed the Containment Assumption and the Structured Negative Assumption.

It is suggested that the Structured Negative Assumption is preferable because it produces a more robust algorithm (POSNEG), which can be easily generalised to the learning of multiple concepts (MULTI). However, the main conclusion of the analysis is that the description space is the crucial factor in the success of concept learning, and that research should be aimed at ways of creating and adjusting the structure of the description space while learning. Such research could be of general use to machine learning, outside the particular domain of concept learning.

1. INTRODUCTION

Focussing is an important technique for learning rules and concepts (Bundy, Plummer and Silver, 1985), and although it is not a complex technique it has received very litle attention in the literature (an exception is Wielemaker and Bundy, 1985). Our interest was stimulated by the fact that a very similar algorithm can be derived from work by Bruner, Goodnow and Austin (1956) as a psychological process of concept learning. Our current research (on intelligent tutoring systems) requires a machine learning technique which has psychological plausibility and therefore focussing was chosen as the relevant technique. The research reported in this paper was intended to develop the focussing algorithm of Bundy et al. into something more closely related to human learning. In fact, the result was a fuller understanding of the algorithm itself, along with some possibilities for psychological improvements. This paper concentrates on the discussion of the focussing algorithm itself.

To begin with, the standard focussing algorithm is described, along with a simple, but typical, example. This leads on to a consideration of the algorithm's main strengths and weaknesses, which give rise to an alternative algorithm (called POSNEG), which seems to be more powerful than the standard. The next section of the paper tries to understand the nature of this extra power, and it presents an alternative perspective on the nature of the description space and the algorithm. This reveals some assumptions underlying both versions of the algorithm, suggesting that both are specific versions of a common focussing algorithm. From these assumptions a third algorithm (MULTI) is suggested, which can handle multiple concepts at once, and which can handle some disjunctive concepts. However, at the end of these discussions it becomes clear that the important part of the success of focussing lies not in the algorithm, but in the description space.

For the sake of clarity one piece of terminology must be defined:- _includes_ is used to refer to concepts including an example (ie. positive examples are included in the concept, and negative examples are excluded). In contrast, _contains_ refers to a concept including all examples defined by some other concept (ie. the concept coloured objects contains the concept red objects).

2. FOCUSSING

Focussing is a concept learning algorithm, which requires positive and negative examples of the concept to be learnt. These examples are expressed as a collection of _values_ of prespecified _attributes_. The algorithm is based on a _description space_, which represents all the possible concepts which may occur; each concept being described as a set of values, one for each attribute. The description space is a set of trees; each tree describing the structure of the values of a particular attribute. The algorithm works by maintaining "markers" on these trees, so as to represent a _maximally general concept_ (_G_), and a _maximally specific concept_ (_S_). The maximally general concept is initially marked by the root nodes of each tree, representing any value on each attribute. This is because some examples are necessary before any concept can be ruled out. Likewise the maximally specific concept is initially marked as having no value on any tree, since nothing more specific can be claimed about the concept. Positive examples cause S to be altered, through generalisation, while negative examples affect G through a process of discrimination. The generalisation process moves the "markers" for S up the tree towards G, such that the concept described by the new "markers" will include the new positive example. This can always be achieved in only one way, and therefore S is always a single concept. However, the discrimination process moves the G 'markers" down the tree such that the negative example is excluded from the concept described by G. This cannot always be done uniquely which means that G has to be described as a set of concepts (with possibly only one member). To avoid confusion S will also be described as a set (of one concept only). When the markers for S and G coincide it is assumed that learning is complete and the concept they describe is the target concept.

Using Holte's (1986) approach to describing such systems, Figure 1 represents focussing as a Learning System, a Performance System and a Performance System Declarative Aspect. The Learning System extracts information from the presented examples, whereas the Performance System classifies unlabelled examples. The Performance System Declarative Aspect is the representation used for passing information between the two parts of the system.

In standard focussing the Learning System performs both generalisation and discrimination, and the Declarative Aspect is two sets, one - S - a singleton and the other - G - with many members. This lack of duality between the treatment of positive and negative examples and within the Declarative Aspect is much emphasised by Bundy et al (1985), and is one of the complications within the standard algorithm. The task of the Performance System is to discover whether an unlabelled example is included in the concept in S (respond "Yes"), or excluded from the concepts in G (respond "No"), or neither of these (respond "Don't know"). This task is complicated by the fact that G may contain more than one concept, in which case the response is "No" only if the example is excluded from all the concepts in G.

2.1. Example 1

Figure 2 displays the description space for all the examples given in this paper. Although this tree is binary, none of the algorithms described depends upon this fact. Clearly it is a very limited space, but it is sufficient to demonstrate the important points. In the following example the concept to be learnt is "any black object". Initially the markers for G are at the top of the tree, and the markers for S are at the bottom. Thus,

G = { <anycolour anyshape> };   S = { <nocolour noshape> }.

(1) Positive example: <black triangle>.

G = { <anycolour anyshape> };   S = { <black triangle> }.

Given this example, the maximally specific concept must be that example. There is no reason for G to be altered.

(2) Negative example: <red circle>.

G = { <monochrome anyshape> <anycolour pointed> };
S = { <black triangle> }.

The maximally general concept must now exclude red circles. This can be done by either excluding red objects from the concept, or circular objects. But G's markers need only move towards S, since eventually S and G must coincide. Thus, G now contains two possible concepts, each of which would lead to a correct classification of the two examples seen so far.

(3) Positive example: <black oval>.

G = { <monochrome anyshape> }; S = { <black anyshape> }.

This example is inconsistent with one of the concepts in G (pointed objects), and therefore this concept is removed from G. Also, S is generalised to include this example.

(4) Unlabelled example: <red triangle>.

This example is not included in the concept in S, nor is it included in the concepts in G and thus, it cannot be a member of the concept being learnt. When the system is then told that "No" was the correct answer, no changes are necessary in G or S.

G = { <monochrome anyshape> }; S = { <black anyshape> }.

(5) Negative example: <white square>.

G = { <black anyshape> }; S = { <black anyshape> }.

Since this example is included in G's concept, it is necessary for G to be altered by discrimination. This produces two possibilities, but only one of which is towards S.

(6) Unlabelled example: <black circle>.

At this point, when G and S are equal, learning is complete and unlabelled examples can be classified without doubt. Since black circles are clearly included in S's concept, the response should be "Yes". Also, it is clear that the concept 'black objects' has been successfully learnt.

2.2. Strengths and Weaknesses of Focussing

The above example illustrates most of the important features of focussing, and from it we can comment on some of the strengths and weaknesses of the algorithm.

2.2.1. Strengths

(1) Learning is incremental. This is different from, for example, Quinlan's ID3 classification algorithm (Quinlan, 1983), which requires all examples to be presented simultaneously.

(2) Also unlike ID3, focussing produces a compact representation of the concept being learnt. The algorithm is not distracted by the presence of irrelevant attributes in the description space.

(3) Memory for all the training instances is not necessary for learning, since all relevant information is extracted from each example when it is presented.

(4) There is always a partially learnt concept which can be used to classify any unlabelled examples. It is possible to describe this concept as just S, or just G or as both (as the system above does), in which case the classification can include "Don't know".

2.2.2. Weaknesses

(1) The algorithm can only learn conjunctive concepts, since disjunctives lead to overgeneralisation and inconsistencies.

(2) The presence of 'far misses' (ie. where discrimination does not lead to a single solution) can lead to a substantial increase in processing load, since they generate a lot of alternatives within G.

(3) There is a lack of duality in the processing of positive and negative examples.

(4) There is no capability for handling noisy data.

(5) The description space, which must be specified in advance of learning, may be inadequate.

2.3. Psychological Issues and Improvements

Of these strengths and weaknesses, it is apparent that most of the strengths are important for a psychological model of concept learning. For example, human learning is clearly incremental, and also it succeeds without storing all of the training examples. People can also make judgments about unknown examples, even when only a few examples have been observed.

Likewise the weaknesses are important. Most of them would be serious weaknesses in a psychological model of concept learning. The last two, however, have proved to be problematic for much of the machine learning research, not just focussing. Some attempts have been made to deal with these problems, but many of them succeed only at the expense of some of the strengths. Here, we will discuss attempts to tackle these problems which do not compromise the strengths of focussing.

The problem of 'far misses' is handled by Bruner et al (1956), in their psychological model, by processing information from positive examples only. In doing this they have taken S as their current concept, and not used G at all. This means that they cannot tell when learning is complete, though it is not clear that this is necessary for a psychological model.

Winston (1975) overcame the problem of 'far misses' by only allowing 'near misses' to be presented as negative examples. Unfortunately, as van Someren (1986) points out, focussing based on 'near misses' alone is inadequate, because the nature of a 'near miss' is dependent upon the structure of the description space. Van Someren's solution is to allow domain knowledge to adapt the description space, in order that previous 'far misses' become 'near misses'. Similarly, Wielemaker and Bundy (1985) tackle the problem of the description space, by adjusting the structure of individual trees. The approach we take here is to reformulate the focussing algorithm itself, in an attempt to solve the problem of both 'far misses' and disjunctive concepts. The discrimination process is, therefore, our major interest and it is this which we propose to adjust.

3. AN ALTERNATIVE APPROACH (POSNEG)

3.1. The Algorithm

Figure 3 displays POSNEG according to Holte's distinctions. The main changes are that the Learning System performs generalisation only, with the Performance System using discrimination. This means that the PSDA is simply two maximally specific concepts, one for the positive examples and one for the negative examples. The genralisation process is identical to that for standard focussing, which means that after each example there is a unique maximally specific concept (positive or negative, depending on the status of the example). The maximally specific concept for the positive examples is referred to as POS, and that for the negative examples is referred to as NEG. In this way there is no distinction between 'far misses' and 'near misses', since they are all summarised as negative examples. Thus, two of the weaknesses of the standard algorithm are tackled together.

The discrimination process is very similar in style to standard focussing, with the difference that it occurs within the Performance System and, instead of discriminating a negative example from the set G, it discriminates POS from NEG. This means that the results of the discrimination process define the concept learnt so far, rather than the current set of maximally general concepts, which have to be stored for comparison with later examples. Thus the task of the Performance System is simply to compare an unlabelled example with the result of discriminating POS and NEG.

3.1.1. The discrimination process

(a) If POS = NEG then no discrimination is possible. There is no description of the concept within the description space (assuming the examples were correctly classified).

(b) If POS does not contain NEG or overlaps with it then POS is the concept. Oveerlapping can occur when POS or NEG becomes overgeneralised. It is assumed that the negative examples are more likely to be the cause of overgeneralisation.

(c) If POS contains NEG then resolve (see d) the values of the first attribute and attach them to the remaining values in POS, and attach the first value in POS to the result of discriminating the remainder of POS and NEG. In this way at least two descriptions of the concept are generated. The Performance System will respond "Yes" if the example is included in any of them (ie. the concept can be disjunctive).

(d) To resolve two values of a particular attribute is to obtain all the nodes of the attribute tree, which are below the positive value and not above the negative (this is like discrimination in the standard algorithm). If there is more than one such node, then they all define the concept for this attribute. This process allows recovery from overgeneralisation.

Having thus performed discrimination, all the performance system has to do is to evaluate whether the unlabelled example is included in any concept in the result. This gives rise to a simple "Yes"/"No" response.

3.2. Example 1 Again

The description space and the concept to be learnt are the same as for the previous example (ie. all black objects).

NEG = <nocolour noshape>; POS = <nocolour noshape>.

(1) Positive example: <black triangle>.

NEG = <nocolour noshape>; POS = <black triangle>.

(2) Negative example: <red circle>.

NEG = <red circle>; POS = <black triangle>.

POS is not altered following a negative example, so this example simply updates NEG.

(3) Positive example: <black oval>.

NEG = <red circle>; POS = <black anyshape>.

(4) Unlabelled example: <red triangle>.

At this point the Performance System is required and discrimination occurs. The result of the discrimination process is the concept <black anyshape>, since POS does not contain NEG and therefore the result of discrimination is POS. Following discrimination the Performance System checks whether the example is included in the result. In this instance it is not and, thus the response is "No". When the system is told that this is indeed the correct answer then NEG will be updated by the now labelled example. Notice that this is the first time that discrimination has been performed, and that the result is not stored.

NEG = <red anyshape>;   POS = <black anyshape>.

(5) Negative example: <white square>

NEG = <anycolour anyshape>; POS = <black anyshape>.

At this point learning is complete as far as negative examples go. Generalisation of the negative examples has the produced the most general concept, and no more information can be obtained from negative examples. Learning will continue when further positive examples are seen.

(6) Unlabelled example: <black oval>.

The result of the discrimination process is <black anyshape>, which includes this example and so the response for this example is "Yes". Since <black anyshape> is the concept to be learnt, learning is also complete for the positive examples. But POSNEG, unlike the standard algorithm, has no way of knowing this.

3.3. Main Differences with Standard Focussing

I have already described the main differences within the algorithm. In this section I shall outline four implications of these differences:-

(1) Because POSNEG is revolutionary, rather than evolutionary, there is a reduction in the potential for high demands on memory. In standard focussing the results of the discrimination process must be stored, because they are part of the Learning System, but in POSNEG, the results of discrimination

are used by the Performance System and then forgotten.

(2) The Performance System compares unlabelled examples with the concepts resulting from the discrimination process. This yields a simple "Yes"/"No" response, whereas standard focussing can respond "Don't know". However, the Performance System could be easily changed to offer all three responses, simply by checking the example against both POS and NEG. Such changes would not affect the learning system.

(3) POSNEG is symmetrical, since positive and negative examples are treated in exactly the same way. POSNEG's performance would not be affected if the negative examples were labelled as positive and the positive negative, but standard focussing would almost certainly fail.

(4) POSNEG does not detect inconsistencies. Since either POS or NEG can become overgeneralised (due to the structure of the description space), legitimate examples may appear to be inconsistent. If there is a genuine inconsistency (eg. due to noisy data) then the discrimination process will fail to work. However, this failure to detect inconsistency at the time of presentation enables POSNEG to be able to learn a greater range of concepts than standard focussing, as in the following example.

3.4. Example 2

Again the description space is as in Figure 2. The example will be given in both POSNEG and the standard algorithm.

G = { <anycolour anyshape> }; S = { <nocolour noshape> }

NEG = <nocolour noshape>; POS = <nocolour noshape>.

(1) Positive example: <red circle>.

G = { <anycolour anyshape> }; S = { <red circle> }.

NEG = <nocolour noshape>; POS = <red circle>.

(2) Positive example: <green oval>.

G = { <anycolour anyshape>; S = { <coloured round> }.

NEG = <nocolour noshape>; POS = <coloured round>.

(3) Negative example: <black triangle>.

G = { <anycolour round> <coloured anyshape> };

S = { <coloured round> }.

NEG = <black triangle>; POS = <coloured round>.

(4) Negative example: <white square>.

G = { <anycolour round> <coloured anyshape> };

S = { <coloured round> }.

NEG = <monochrome pointed>; POS = <coloured round>.

(5) Positive example: <red triangle>.

G = { <coloured anyshape> }; S = { <coloured anyshape> }.

NEG = <monochrome pointed>; POS = <coloured anyshape>.

At this point the standard algorithm will terminate because G and S are equivalent, the concept being <coloured anyshape>. The result of POSNEG's discrimination process is also

<coloured anyshape>, since this excludes NEG.

(6) Positive example: <white oval>.

The standard algorithm fails at this point, because the example is not contained within S, but POSNEG simply assimilates the information into POS.

NEG = <monochrome pointed>; POS = <anycolour anyshape>.

The discrimination process finds that POS contains NEG and the first two values are 'resolved', giving the result "coloured". One result of the discrimination process is thus <coloured anyshape>, whilst the other depends upon the resolution of "pointed" and "anyshape". The result of this 'resolve' leads to the other result of the discrimination process, which is <anycolour round>. Thus, the concept being learnt is {<coloured anyshape><anycolour round>}, ie. any object which is either coloured or round - a disjunctive concept.

Thus, POSNEG is capable of learning disjunctive concepts, which cannot be handled by the standard algorithm. Since the two algorithms contain basically the same components, it is interesting to wonder exactly why this difference arises - it is not clear why postponing the discrimination process should lead to greater power. In order to understand exactly why this is the case, it is necessary to reexamine our representation of the description space.

4. THE STRUCTURE OF THE DESCRIPTION SPACE

4.1. A spatial representation

The tree structures which are commonly used to represent the description space are not the only way of representing it. In fact, using tree structures obscures some important aspects of the focussing algorithm. In this section of the paper the description space will be described not as a tree, but as a physical space, in which the area occupied by positive and negative examples is explicitly represented. Thus, Figure 4 gives an example of the way the description space could look, with a clear boundary around all conceivable objects, and subsets within that representing the positive examples, the negative examples and those contained within {MGC}.

In both algorithms S and POS are equivalent, since they both summarise the area occupied by positive examples. However, whereas NEG summarises the area of negative examples, G summarises its complement, the area not occupied by negative examples. When examined in this way, it is hard to understand why the two algorithms should produce different results, since it would appear that from NEG it should always be possible to calculate G. The reason for the difference, however, lies in the fact that the space in Figure 4 is unstructured; once structure is imposed then assumptions must be made about that structure, since it is within this structure that generalisation and discrimination occur. Each of the two algorithms described above makes a different assumption about how to utilise this structure, and hence gives different results. Without either of these assumptions the algorithms would, in fact, be the same.

Figure 5 displays a structured space for the description space of Figure 2, indicating the existence of 16 possible objects. Within this space we can mark the area of S, or POS, (and use a '+' to indicate an actual observed example) and we can also mark G, or NEG, using a '-' to indicate an observed negative example. The concepts to be learnt are the rows and columns of the space (and combinations thereof). It should be emphasised that this is purely an illustrative representation, since the internal representation of the description space is exactly the same. Figure 6 illustrates example 2 in this form, using the standard algorithm. Initially, G is the whole space, and S is no part of the space. The first positive example causes S to expand (through generalisation), and the negative examples cause G to shrink (through discrimination). In this way it is possible to see more clearly the significance of a discrimination which generates two or more concepts within G (see Figure 6d). For clarity the two possibilities are labelled G1 and G2, the former being necessary to exclude black objects and the latter to exclude triangular objects. These two possibilites are exclusive and the focussing algorithm wil use the next examples to try and distinguish them. In Figure 6f a positive example serves to reject G2, even though no monochrome round objects have been observed. Thus, a single example can cause large changes in G and S. This fact prevents the algorithm from coping with the next example, which is monochrome and round. The reason that this occurs is that the standard algorithm has made the assumption that G must always contain S, and therefore when a positive example is observed which is not included in G2 (Fig. 6f), G2 can be rejected. I have termed this assumption, which is not essential, the "Containment Assumption".

The Containment Assumption states, quite reasonably, that any maximally general concept must contain the maximally specific concept. But it is this assumption that decrees that focussing can only learn conjunctive concepts, since the assumption only holds true for such concepts.

Figure 7 illustrates the same example for POSNEG. It reveals that this algorithm depends upon a different assumption, one which I have termed the "Structured Negative Assumption". This assumes that the negative of the concept being learnt will be structured similarly to the concept itself and, thus, that generalisation can be applied to the negative examples. Thus, in figure 7f, monochrome round objects are simply not known about - they have not yet been accepted into POS, or rejected by being in NEG. This assumption is in contrast to the standard algorithm which does not consider the negative of the concept.

However, it also becomes apparent from Figure 7 that POSNEG does not really learn disjunctive concepts. Instead, it is learning the reverse of a conjunctive concept. This reveals why the presence of duality is an important aspect of focussing. For the standard algorithm it is very important which are the positive and which are the negative examples, whereas it makes no difference to the symmetrical POSNEG. This suggests that, of the two assumptions, the Structured Negative Assumption may be preferable, since it produces a more robust algorithm.

It is important to realise that if the assumptions are dropped then the performance of the two algorithms is the same. The standard algorithm will generate a set G which contains all concepts except the given negative examples, since it cannot use S to restrict it. POSNEG can only function without its assumption if it simply remembers all negative examples given so far. Thus the two algorithms reduce to the idealised representation of Figure 4. However, without any assumptions the concept learning reduces to little more than a memory task, and no predictions could be made about unlabelled examples.

It has already been suggested that the Structured Negative Assumption may be preferable, because it produces a symmetrical algorithm. A further reason for it to be preferred is that it can be generalised to different learning contexts.

5. MULTIPLE FOCUSSING (MULTI)

Hunt, Marin and Stone (1966) comment how there are in fact very few negative instances of concepts, since almost everything is a positive example of some other concept. Whilst this may not be true for some rule-learning programs, it does seem to be a valid comment about concept learning. The significance of this is that whilst standard focussing can only handle conjunctive concepts, and POSNEG can only manage with conjunctive positives or conjunctive negatives, the Structured Negative Assumption can be applied to a disjunctive concept with a disjunctive negative, if the negative can be split into two or more conjunctive concepts. For the purposes of what follows it will be assumed that the negative concept(s) are naturally defined ones, which are given. It may be possible to develop techniques whereby different divisions of negative instances can be calculated automatically. (*)

This is similar to the use of rule shells for learning disjunctive concepts (as discussed by Bundy et al, 1985). The main difference, however, is that it is the negative examples which are split into two or more groups, not the positive examples. Also, the groups are labelled from the beginning of learning, rather than from the point where a contradiction occurs. The following program assumes the examples to be correctly labelled.

5.1. Example 3

Using the same description space again, consider the three concepts,

(1) Monochrome, round objects or red objects.

(2) Green objects.

(3) Monochrome, pointed objects.

Both algorithms described so far would fail, if they were required to learn concept 1. The standard algorithm would fail because of inconsistencies and POSNEG would fail because both POS and NEG would be <anycolour anyshape>, which would prevent successful discrimination.

However, POSNEG can be adjusted slightly to handle multiple concepts. The algorithm MULTI can be given labelled examples of all

(*) For instance, it may be possible to use some measure which compares the distance of the current example from both the current negative concept and the positive concept. If the example is nearer the positive concept, then a new negative class is begun, otherwise the example is allocated to the nearest negative concept.

three concepts, and successfully learn all three concepts. However, to achieve this it is necessary that the concepts are both exclusive and exhaustive, since this is an implicit assumption of the discrimination process. The only adjustment necessary for the discrimination process is that instead of discriminating POS with NEG, it must discriminate the concept of interest with each of the negative concepts. This is done by first discriminating with one of them, then discriminating the result of this with another, and the result of this with another etc. etc. The resultant list of concepts represents disjunctively the concept of interest.

Consider the following sequence of training examples. Initially CON1, CON2 and CON3 are all equal to <nocolour noshape>.

(1) Example of concept 1: <red square>.
CON1 = <red square>; CON2 = <nocolour noshape>; CON3 = <nocolour noshape>.

(2) Example of concept 3: <white triangle>.
CON1 = <red square>; CON2 = <nocolour noshape>; CON3 = <white triangle>.

(3) Example of concept 3: <black square>.
CON1 = <red square>; CON2 =<nocolour noshape>; CON3 = <monchrome pointed>.

(4) Example of concept 1: <white circle>.
CON1 = <anycolour anyshape>; CON2 = <nocolour noshape>; CON3 = <monochrome pointed>.

(5) Example of concept 2: <green circle>.

CON1 = <anycolour anyshape>; CON2 = <green circle>; CON3 = <monochrome pointed>.

(6) Example of concept 2: <green square>.

CON1 = <anycolour anyshape>; CON2 = <green anyshape>; CON3 = <monochrome pointed>.

Note how as with POSNEG none of the inconsistencies has been detected; this is because no discrimination has occurred. To describe each concept (or to classify an unlabelled example) then discrimination must occur. For concepts CON2 and CON3 this results in themselves being the concept, since they each overlap with, or do not contain the other two concepts. However, CON1 can only be described through the 'resolve' operation. Initially MULTI discriminates CON1 and CON2 which gives the result {<red anyshape> <monochrome anyshape>}, and then each of these is discriminated with CON3, giving the result {<red anyshape><monochrome round>}.

Thus, MULTI can learn more complex disjunctive concepts than POSNEG can. Even so it depends upon the existence of only 1 disjunctive concept, since more than one would lead to overgeneralisation on both of them, and the discrimination process would be unable to distinguish which values of which attributes discriminate the two. However, it is important to realise that this generalisation is only possible with the Structured Negative Assumption, and not with the Containment Assumption, because of the symmetricality of the algorithm. But, apart from the difference in duality, there is not much difference between the algorithms. They use the supplied structure of the description space to generalise and discriminate. Neither of these processes is particularly complicated, and what the spatial representation of the description space makes clear is that the real determinant of the success of focussing is the structure of the description space itself.

6. SUMMARY AND IMPLICATIONS

This paper has presented an alternative focussing algorithm, and it has demonstrated that there is a core algorithm underlying both, which simply compares the positive examples seen with all possible positive examples, given the observed negative examples. The standard algorithm and POSNEG (and MULTI, too) make assumptions about the relationship between the structure of the description space and the algorithm, in order to prevent the brute force search of all possibilities.

However, the Containment Assumption used by the standard algorithm produces a program which lacks duality, which can only handle conjunctive concepts and which cannot be generalised easily. By contrast, the Structured Negative Assumption produces a symmetrical algorithm, which handles a small set of disjunctive concepts, and which can be generalised to the learning of more than one concept.

By discussing small examples, and representing the description space as an n-dimensional space, it has become clear that the focussing methods discussed depend upon the existence of conjunctive concepts, even when the main concept is not conjunctive. This may be true for the two assumptions discussed here, but there may be others to which it does not apply. An area of interesting research would be to examine other possible ways of constraining the search for the concept.

However, the main implication of this analysis is the overwhelming importance of the structure of description space. The set of concepts which can be learnt by these algorithms depends crucially on whether the given structure maps onto the structure within the concept. Although the new assumptions described here make the algorithms slightly more powerful, the only way to dramtically concept learning performance is to be able to change, dynamically, the structure of the description space. This is being attempted by Wielemaker and Bundy (1985), and by van Someren (1986). However, their approaches are only to be used when inconsistencies occur in the example set, whereas it would seem desirable to find a way of checking for patterns in the examples.

In much of human learning the relevant structure of an attribute's values has to be inferred from the examples. For example, people cope well with concepts which involve numbers, even though such an attribute cannot be readily represented as a tree structure. People can structure a numerical attricute into primes, even numbers, multiples of <n>, etc. etc. An important question, therefore, for machine learning is to discover techniques for dynamically varying the structure of the description space, and to discover whether there are any constraints (psychological or otherwise) on the restructurings which can be performed.

ACKNOWLEDGMENTS

I would like to thank John Self, Stephen Payne and Louise O'Callaghan for their fruitful discussions on the problems of concept learning, both by machines and by people. This research was carried out under a UK Science and Engineering Research Council grant, number GR/D/16079.

REFERENCES

Bruner, J.S., Goodnow, J.A. and Austin, G.A. (1956) A Study Of Thinking, Wiley, New York.

Bundy, A., Silver, B. and Plummer, D. (1985) "A review of rule-learning programs." Artificial Intelligence, 27, 137 - 181.

Holte, R.C. (1986) "Alternative information structures in incremental learning systems." Paper presented at the European Working Session on Learning, Orsay, Paris, January, 1986.

Hunt, E.B., Marin, J. and Stone, P.J. (1966) Experiments In Induction, Academic Press, New York.

Quinlan, J.R. (1983) "Learning efficient classification procedures and their application to chess end games." In Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (Eds.) Machine Learning: An Artificial Intelligence Approach, Tioga Pub. Co., Palo Alto.

van Someren, M.W. (1986) "Constructive induction rules: reducing the description space for rule learning." Paper presented at the European Working Session on Learning, Orsay, Paris, January, 1986.

Wielemaker, J. and Bundy, A. (1985) "Altering the description space for focussing." Paper presented at Expert Systems - 85, Warwick, England.

Winston, P.H. (1975) "Learning structural descriptions from examples." In Winston, P.H. (Ed) The Psychology Of Computer Vision, McGraw-Hill.

Figure 1: Standard focussing algorithm using Holte's
          (1986) representation.

P 30

```
 _______________________________
| LEARNING SYSTEM (LS)          |
|-------------------------------|
|                               |
|    ---- Generalisation <----+----  Positive examples
|    |                          |
|    |                          |
|    |                          |
|    |                          |
|    |    Discrimination <----+----  Negative examples
|    |            |             |
|    |            |             |
|    |            |             |
+----+------------+-------------+
     |            |
     |            |
     V            V
    MSC        { MGC }   =   PERFORMANCE SYSTEM
                             DECLARATIVE ASPECT (PSDA)
     |            |
     |            |
+----+------------+-------------+
|    |            |             |
|    |            |       <-----+-----  Unlabelled examples
|    |            |             |
|    V            V             |
|                               |
|  Below MSC?  Above {MGC}?     |
|                               |
|                ---------------+----> Response
|                               |
|_______________________________|
| PERFORMANCE SYSTEM (PS)       |
+-------------------------------+
```

Figure 2: The Description Space used throughout this paper.

```
             anycolour                               anyshape
               /  \                                    /  \
              /    \                                  /    \
             /      \                                /      \
      coloured      monochrome                 pointed    round
        /  \          /  \                       /  \       /  \
       /    \        /    \                     /    \     /    \
     red   green   black  white          triangle  square  oval circle
      |     |        |      |                |        |       |     |
      |     |        |      |                |        |       |     |
       ------nocolour------                   ------noshape------
```

Figure 3: The POSNEG focussing algorithm using Holte's (1986) representation.

```
 ______________________________
| LEARNING SYSTEM (LS)         |
|------------------------------|
|                              |
|                       <----+----   Positive examples
|   Generalisation             |
|                       <----+----   Negative examples
|   |            |             |
|   |            |             |
+---+------------+-------------+
    |            |
    |            |
    V            V
   POS          NEG     =   PERFORMANCE SYSTEM
                            DECLARATIVE ASPECT (PSDA)
    |            |
    |            |
+---+------------+-------------+
|   |            |             |
|   |            |      <----+-----  Unlabelled examples
|   V            V             |
|                              |
|   Discrimination             |
|         &            ---------+---->  Response
|     Evaluation               |
|                              |
|______________________________|
| PERFORMANCE SYSTEM (PS)      |
+------------------------------+
```

Figure 4: A spatial representation of the description space. (The outer box contains all possible objects in the space.)

```
+---------------------------------------+
|                                       |
|                     +------+          |
|                     | -ve  |          |
|                     | egs  |          |
|                     +------+          |
|                                       |
|    +------+                           |
|    | +ve  |        {MGC}              |
|    | egs  |                           |
|    +------+                           |
|                                       |
+---------------------------------------+
```

Figure 5: An alternative representation of the description space of figure 2.

P 34

```
                            anyshape
                       pointed    round
                       squ tri   ovl cir
                     +-------------------+
                     |    |    |    |    |
             red     |    |    |    |    |
      coloured       +-------------------+
             green   |    |    |    |    |
                     |    |    |    |    |
anycolour            +-------------------+
                     |    |    |    |    |
             black   |    |    |    |    |
      monochrome     +-------------------+
             white   |    |    |    |    |
                     |    |    |    |    |
                     +-------------------+
```

Figure 6: An illustration of the description space during
example 2 for the standard algorithm.

P 35

```
             squ tri  ovl cir                            s   t     o   c
   (a)  +-------------------+                   (b)  +-------------------+
        |    |    |    |    |                        |    |    |    | +  |
  red   | G  | G  | G  | G  |                     r  | G  | G  | G  | S  |
        +-------------------+                        +-------------------+
 green  |    |    |    |    |                     g  |    |    |    |    |
        | G  | G  | G  | G  |     positive           | G  | G  | G  | G  |
        +-------------------+     ------->           +-------------------+
        |    |    |    |    |       red              |    |    |    |    |
 black  | G  | G  | G  | G  |      circle         b  | G  | G  | G  | G  |
        +-------------------+                        +-------------------+
 white  |    |    |    |    |                     w  |    |    |    |    |
        | G  | G  | G  | G  |                        | G  | G  | G  | G  |
        +-------------------+                        +-------------------+


             s    t    o   c                             s    t    o   c
   (c)  +-------------------+                   (d)  +-------------------+
        |    |    |    | +  |                        |    |    |    | +  |
     r  | G  | G  | S  | S  |                     r  | G1 | G1 | S  | S  |
        +-------------------+                        +-------------------+
     g  |    |    | +  |    |                     g  |    |    | +  |    |
  Pos.  | G  | G  | S  | S  |        Neg.            | G1 | G1 | S  | S  |
 ----->  +-------------------+      ------->          +-------------------+
 green  |    |    |    |    |       black            |    | _  |    |    |
  oval b| G  | G  | G  | G  |      triangle       b  |    |    | G2 | G2 |
        +-------------------+                        +-------------------+
     w  |    |    |    |    |                     w  |    |    |    |    |
        | G  | G  | G  | G  |                        |    |    | G2 | G2 |
        +-------------------+                        +-------------------+


             s    t    o   c                             s    t    o   c
   (e)  +-------------------+                   (f)  +-------------------+
        |    |    |    | +  |                        |    | +  |    | +  |
     r  | G1 | G1 | S  | S  |                     r  | S  | S  | S  | S  |
        +-------------------+                        +-------------------+
     g  |    |    | +  |    |                     g  |    |    | +  |    |
  Neg.  | G1 | G1 | S  | S  |        Pos.            | S  | S  | S  | S  |
 -----> +-------------------+      ------->          +-------------------+
 white  |    | _  |    |    |        red             |    | _  |    |    |
 squ.  b|    |    | G2 | G2 |      triangle       b  |    |    |    |    |
        +-------------------+                        +-------------------+
     w  | _  |    |    |    |                     w  | _  |    |    |    |
        |    |    | G2 | G2 |                        |    |    |    |    |
        +-------------------+                        +-------------------+


             s    t    o   c
   (g)  +-------------------+
        |    | +  |    | +  |
     r  | S  | S  | S  | S  |
        +-------------------+
     g  |    |    | +  |    |
  Pos.  | S  | S  | S  | S  |        Following this last positive example
 -----> +-------------------+        it is clear that the example is not
 white  |    | _  |    |    |        contained within S.
 oval  b|    |    |    |    |
        +-------------------+
     w  | _  |    | +  |    |
        |    |    |    |    |
        +-------------------+
```

Figure 7: An illustration of the description space during example 2, for POSNEG.

P 36

```
           squ tri   ovl cir                          s    t    o    c
     (a) +-------------------+                  (b) +-------------------+
         |    |    |    |    |                      |    |    |    | +  |
     red |    |    |    |    |                    r |    |    |    | P  |
         +-------------------+                      +-------------------+
   green |    |    |    |    |                    g |    |    |    |    |
         |    |    |    |    |       positive       |    |    |    |    |
         +-------------------+       ------->       +-------------------+
         |    |    |    |    |         red          |    |    |    |    |
   black |    |    |    |    |        circle      b |    |    |    |    |
         +-------------------+                      +-------------------+
   white |    |    |    |    |                    w |    |    |    |    |
         |    |    |    |    |                      |    |    |    |    |
         +-------------------+                      +-------------------+
```

```
             s    t    o    c                         s    t    o    c
     (c) +-------------------+                  (d) +-------------------+
         |    |    |    | +  |                      |    |    |    | +  |
       r |    |    | P  | P  |                    r |    |    | P  | P  |
         +-------------------+                      +-------------------+
       g |    |    | +  |    |                    g |    |    | +  |    |
 Pos.    |    |    | P  | P  |         Neg.         |    |    | P  | P  |
 ----->  +-------------------+       ------->       +-------------------+
 green   |    |    |    |    |         black        |    | -  |    |    |
  oval b |    |    |    |    |       triangle     b |    | N  |    |    |
         +-------------------+                      +-------------------+
       w |    |    |    |    |                    w |    |    |    |    |
         |    |    |    |    |                      |    |    |    |    |
         +-------------------+                      +-------------------+
```

```
             s    t    o    c                         s    t    o    c
     (e) +-------------------+                  (f) +-------------------+
         |    |    |    | +  |                      |    | +  |    | +  |
       r |    |    | P  | P  |                    r | P  | P  | P  | P  |
         +-------------------+                      +-------------------+
       g |    |    | +  |    |                    g |    |    | +  |    |
 Neg.    |    |    | P  | P  |         Pos.         | P  | P  | P  | P  |
 ----->  +-------------------+       ------->       +-------------------+
 white   |    | -  |    |    |          red         |    | -  |    |    |
 squ.  b | N  | N  |    |    |       triangle     b | N  | N  |    |    |
         +-------------------+                      +-------------------+
       w | -  |    |    |    |                    w | -  |    |    |    |
         | N  | N  |    |    |                      | N  | N  |    |    |
         +-------------------+                      +-------------------+
```

```
             s    t    o    c
     (g) +-------------------+
         |    | +  |    | +  |
       r | P  | P  | P  | P  |
         +-------------------+
       g |    |    | +  |    |
 Pos.    | P  | P  | P  | P  |
 ----->  +-------------------+
 white   |    | -  |    |    |
 oval  b | N  | N  | P  | P  |
         +-------------------+
       w | -  |    | +  |    |
         | N  | N  | P  | P  |
         +-------------------+
```

Following this last positive example the set POS extends to the whole space except NEG, thus learning the concept.

DISCOVERY OF ALGORITHMS FROM WEAK METHODS

Armand E. Prieditis

Computer Science Department
Rutgers University
New Brunswick, NJ 08903
ARPAnet address: PRIEDITIS@RUTGERS

July 1986

Keywords: algorithm discovery, explanation-based generalization, goal regression, learning of algorithms, planning, problem-solving.

Abstract

Weak problem-solving methods (e.g. means-ends analysis, breadth-first search, best-first search) all involve a *search* for some sequence of operators that will lead from an initial state to a goal state. This paper shows how it is possible to learn operators whose bodies contain algorithmic control constructs (e.g. loops, sequences, conditionals) such that the control construct itself applies the sequence needed to lead from the initial state to a goal state *without* a search for the sequence. By using explanation-based generalization (Mitchell, 1986) and an explicit theory of algorithms, the method learns operators (whose bodies contain algorithmic control constructs) that represent logically valid generalizations of the solution sequence.

1. Introduction

What are algorithms and where do they come from? How are algorithms discovered and learned from weak problem-solving methods? This paper presents a method for learning compound operators[1] whose bodies contain explicit algorithmic control constructs such as sequences, conditionals, and loops (expressed here as tail recursion) from single examples of solution sequences in the blocks-world domain. Explanation-based generalization[2] and a theory of algorithms provides a logically valid basis for the learning. This section first gives a general statement of the problem and describes why current methods were inadequate to deal with learning complex algorithmic control constructs. Next, the algorithm learning problem is presented in table 1-1 along with a brief discussion of

[1]The operators shown in this paper use a STRIPS-style (Fikes, et al., 1972) representation along with an operator body. The appendix lists the four primitive operators in the blocks world domain.

[2]See (Mitchell, 1986, DeJong, 1983, Minton, 1984, Mahadevan, 1985, Utgoff, 1984) for more detailed discussions and comparisons of the term "explanation-based generalization".

notation used and how the algorithm learning problem relates to explanation-based generalization. This section concludes with an outline of the general method for the discovery of algorithms.

The rest of the paper is organized as follows. Section 2 presents one particular algorithm theory (that of repeated loops represented as tail recursion) and examines algorithm discovery and learning in the task of destroying a particular tower composed of three blocks. The example is interesting from another perspective since it also demonstrates the learning and representation of a complex concept: a tower of any number of blocks. Finally, section 3 discusses some future directions, limitations and summarizes the paper.

1.1. Statement of the Problem

This research addresses the problem of learning operators whose bodies contain algorithmic control constructs that represent a generalization of the solution sequence obtained from a weak method. Machine learning and automatic programming research have each examined learning general algorithmic control constructs from examples traces of problem-solving behavior from different perspectives.

In the machine learning research on learning algorithmic macro-operators (or chucks)[3] that represent compilations of sequences, two problem arise. First, though the macro-operators do have bodies that contain algorithmic control constructs which represent solution sequences (either explicitly or implicitly as is the case of a chunk), they could never contain other more complex control constructs such as loops (or a chunked concept representing an arbitrary number of particular chunks that repeat) since the learning systems lacked an explicit theory of other control constructs.

Second, as mentioned above, the learning systems do have a theory of sequences so that algorithmic constructs that represent sequences can be learned, but the theory is implicitly embedded into the learning system's procedures under the assumption that *only* sequences will be learned. So, while a macro-operator such as **unstack-putdown**(x,y) (below) could be learned, operators representing repeated application of **unstack-putdown** an arbitrary number of times could not be learned.

```
unstack-putdown(x,y)
    P: {HANDEMPTY,CLEAR(x),ON(x,y)}
    D: {ON(x,y)}
    A: {CLEAR(y),ONTABLE(x)}
    BODY: unstack(x,y),
          putdown(x).
```

Automatic programming literature,[4] on the other hand, describes programs that did induce more complex algorithmic control constructs (such as loops) from example traces, but the methods described lacked the logically valid basis for generalizing from the examples and typically relied on many training examples. Explanation-based generalization (EBG) and the algorithm theory together provide the logically valid basis for generalizing from single examples of solution traces and the type of inductive generalization that occurs in

[3]See (Korf, 1985, Mahadevan, 1985, Laird, et al., 1984) for excellent examples of macro-operator learning, learning compilations of proofs, and chunking respectively.

[4]See (Bauer, 1977, Phillips, 1977, Siklossy, 1977) for some examples.

automatic programming.

Explanation-based generalization provides the justification for considering only those parameters in an operator that are important for achieving a general goal and the algorithm theory describes, in general, how new operators which contain algorithmic control constructs in their bodies are constructed. Derived from standard methods used in algorithm verification and proving programs correct (Hoare, 1973, Manna, 1974, Mili, 1985), the theory represents in essence compiled knowledge about the proofs used to verify various algorithmic constructs *in general*.

Table 1-1 summarizes the general algorithm learning problem with specific references to the example that will be shown in section 2. The table roughly corresponds to that of (Mitchell, 1986)--which states the general EBG problem for integral calculus. The similarities and differences between (Mitchell, 1986) and the table are discussed below.

- Here the goal concept, CONSTRAINTS(x,s), is a class of constraints on an object, x, and on a state s, such that application of a sequence will lead to the general goal. In (Mitchell, 1986 the goal concept was "when it is useful to apply a particular operator" such that it will lead to a general goal. The particular operator was the first in a sequence of operators leading to the general goal. As will be shown later in section 2, the object x referenced in the goal concept allows for justifiable parameterization.

- As in (Mitchell, 1986) the domain theory relates the predicates SOLVABLE and SOLVED to features that are directly testable in the problem state: in this case HANDEMPTY, ON, and CLEAR. SOLVED and SOLVABLE are predicates that embody some knowledge about search.

- Here algorithm theory and operators are part of the explicit inputs.

- Here the goal is to discover new operators that contain algorithmic control constructs in their bodies and not search control heuristics as in (Mitchell, 1986). Note that search control heuristics on each of the operators in a sequence may actually make the problem solver apply the entire sequence *in sequence* as would a single algorithm which contained the control structures needed to execute the exact sequence, except that there is no guarantee the problem solver would make all of the correct choices under the time or memory constraints of the problem-solver. In fact, in the case of repeated operator application, there is no deterministic knowledge the problem-solver could use to apply the operator repeatedly without considering other potential operators.

1.2. The General Method

This section describes the general method in five steps and then examines the five steps in more detail. The five steps will be shown again for the particular example in section 2.

Table 1-1: THE ALGORITHM LEARNING PROBLEM

Given:

- *Goal Concept:* A class of constraints about an object, x, on a state, s, such that CONSTRAINTS(x,s) holds. That is, the set of states that will lead to a given goal by a set of algorithm applications. The particular goal concept used in the example is CONSTRAINTS(C,0).

  CONSTRAINTS(x,s) ⇔ NOT(SOLVED(x,s)) ∧ SOLVABLE(x,s)

- *Operators:* A set of the currently known problem-solving operators such as **pickup**, **putdown**.

- *Training Example:* The initial state is referenced by the number 0, where HOLDS({ON(A,B),ON(B,C),CLEAR(A),ONTABLE(C),HANDEMPTY},0).

- *Domain Theory:*

  SOLVABLE(x,s) ⇔ (∃op)(SOLVED(x,result(op,s)) ∨
     SOLVABLE(x,result(op,s)))

  (∃y)HOLDS({ON(y,x)},s) ⇒ NOT(SOLVED(x,s))
  HOLDS({CLEAR(x),HANDEMPTY},s) ⇒ SOLVED(x,s)
  HOLDS(z,apply(op,s)) ⇔ HOLDS(REGRESS(z,op),s)

- *Operationality Criterion:* The concept must be expressed in terms of HOLDS(x,s) where x is a description of a state using predicates recognized by the problem-solver (e.g. ON, CLEAR, HANDEMPTY) and s is the initial state (in this case 0).

- *Algorithm Theory:* A theory that describes the semantics of algorithms. An example of a particular theory is one that describes the semantics of repeating operator application in terms of the operator that repeats and the argument structure between any two consecutive instances of the repeating operator.

Discover:

- *New Operators:* Operators that contain algorithmic control constructs in their bodies along with the parameters to the algorithm. The bodies are a generalization of the sequence of operators that was applied in the training example. The preconditions of the operator are sufficient to guarantee the general goal (in this case HOLDS({CLEAR(x),HANDEMPTY},s) after the operator application. In the example the parameter is x.

Given the inputs described in table 1-1, the general method for discovering and learning algorithms can be summarized by the following steps:

STEP 1. *Restate* the goal concept by finding the weakest preconditions for a sequence of operators such that the specific goal will still hold.

STEP 2. *Discover* algorithmic control constructs in the solution sequence.

STEP 3. *Find* the specialization of the discovered operator's preconditions such that it implies the weakest precondition from step one.

STEP 4. *Regress* the general goal concept through the discovered operator's preconditions with the bindings that specialized the discovered operator's preconditions.

STEP 5. *Instantiate* the discovered operator with the generalization obtained from step four.

Step 1 corresponds to EBG without the final step of regressing the general goal concept through the explanation structure. As shown in table 1-1 the goal concept specifies the object or objects (x), mentioned in the goal state and the state (s). Given that the goal state specifies constraints[5] on specific object or objects (x), the weakest precondition is expressed as a *general class of constraints* on the objects or object (x) mentioned in the goal and a state (s).[6] The result is a statement that shows the general class of constraints on an object implies the goal concept.

Step 2 uses an algorithm theory that contains template-like knowledge about how to build new operators from existing ones--that is, how more complex operators are semantically defined in terms of simpler ones. The existing operators appear in the solution sequence from the explanation structure.

In step 3 it is important to recall that the general class of constraints from step 1 represents the weakest precondition of the sequence *with respect to* a general goal. The specialization, in the form of variable bindings in the discovered operator's preconditions to objects mentioned in the goal concept (x), tunes the algorithm to the general goal--in this case HOLDS({CLEAR(x).HANDEMPTY}.s).

Step 4 provides the logical justification for parameterizing the discovered operator according to the goal. From step 2, the discovered operators certainly did have parameters, however, they were not tuned to the general goal.

Step 5 simply adds the new operator to the existing set of operators. The parameters to the new operator are the objects or object referenced in the goal concept (x).

This section has shown how current learning techniques were inadequate in two ways for learning complex control constructs other than sequences. One inadequacy stemmed from not having an explicit theory of the algorithmic constructs over which the generalization

---

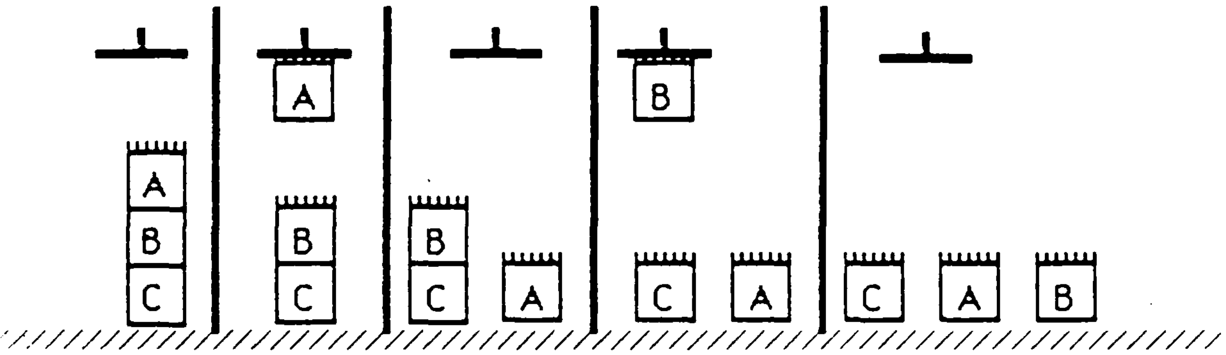
[5]An example of a constraint on an object C is clear(C).

[6]The predicate HOLDS(u.s) is used to denote a description u that is true in a state s. The initial state is referenced by 0. For example, HOLDS({clear(C)},0) states that object C is clear in the state referenced by 0. All other states are the result of applying some operator to a state (i.e. HOLDS(x.result(op.s)) states that x is true as a result of applying action a to s).

was occurring. The other inadequacy was lack of logical justification for inducing certain algorithmic constructs. A method was presented that showed how the inadequacy could be dealt with. The general algorithm learning problem--it's inputs and outputs--was presented along with a discussion of the general method. The next section re-examines each of the five steps of the general method in a specific example.

2. Examples

Figure 2-1: THE SEQUENCE OF OPERATORS APPLIED IN THE EXPLANATION OF CONSTRAINTS(C,0)

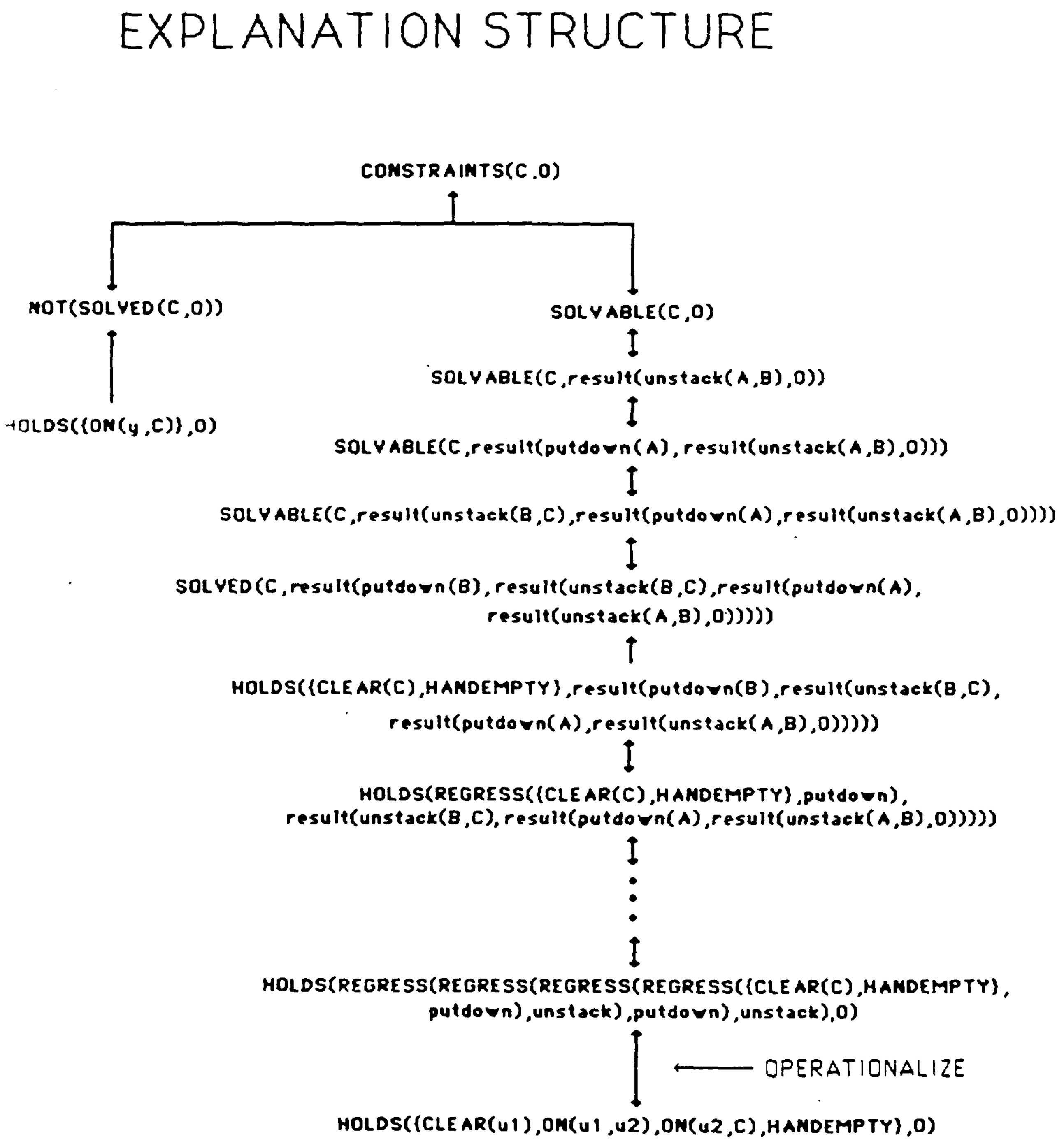


This section presents one particular algorithm theory and demonstrates how an operator is learned from a single example. The particular training example consists of the left-most state (denoted by 0) shown in figure 2-1. Blocks with the bristles on top of them represent the notion of CLEAR. The exact sequence of operators is shown in the explanation of CONSTRAINTS(C,0) in figure 2-2. The resulting algorithmic construct is one which unstacks all the blocks from a given block. The rest of this section consists of a brief explanation of the general algorithm theory used and is followed by a trace of the five steps outlined in section 1. The section concludes with a short discussion of the learning process.

The general algorithm theory of a looping construct is shown in the top of figure 2-3. The theory can be viewed as a template that instantiates a new operator given existing operators and the general pattern of arguments between any two consecutive instances of existing operators. For purposes of presentation. the template is shown in boldface and the arguments that are evaluated are in italics so that the general theory of the looping construct can later be compared to the middle of figure 2-3--the portion that represents the particular instantiation that will be learned from the example. The operator that actually repeats is included in the bottom of figure 2-3 for reference.

Once an operator has been found that repeats at least twice in the solution sequence. the general patterns of two consecutive instances of its arguments can be obtained easily from the explanation structure. Suppose. as in the example, that the general pattern of of any two consecutive instances of an operator called **unstack-putdown** is (u1,u2) and (u2,u3). The algorithm theory claims that a new operator can be instantiated that represents the repeating version of **unstack-putdown** (called **r-unstack-putdown**) with a certain list

Figure 2-2: EXPLANATION OF CONSTRAINTS(C,0)


EXPLANATION STRUCTURE
CONSTRAINTS(C,0)
NOT(SOLVED(C,0))
HOLDS({ON(y,C)},0)
SOLVABLE(C,0)
SOLVABLE(C,result(unstack(A,B),0))
SOLVABLE(C,result(putdown(A),result(unstack(A,B),0)))
SOLVABLE(C,result(unstack(B,C),result(putdown(A),result(unstack(A,B),0))))
SOLVED(C,result(putdown(B),result(unstack(B,C),result(putdown(A), result(unstack(A,B),0)))))
HOLDS({CLEAR(C),HANDEMPTY},result(putdown(B),result(unstack(B,C), result(putdown(A),result(unstack(A,B),0)))))
HOLDS(REGRESS({CLEAR(C),HANDEMPTY},putdown), result(unstack(B,C),result(putdown(A),result(unstack(A,B),0))))
HOLDS(REGRESS(REGRESS(REGRESS(REGRESS({CLEAR(C),HANDEMPTY}, putdown),unstack),putdown),unstack),0)
OPERATIONALIZE
HOLDS({CLEAR(u1),ON(u1,u2),ON(u2,C),HANDEMPTY},0)

structure as its argument.[7]

The list structure represents a compaction of common objects referenced throughout the looping construct. The list structure is built by noting the common portions of the two argument instances *args1* and *args2* (in the above case (u1,u2) and (u2,u3)) and by combining them into one (u1,u2,u3) so that any given triple in the list structure can be decomposed again into the original arguments. Some simple sorting permutation is done to get matching arguments right-most in *args1* and left-most in *args2*. The sorted version of *args1* is denoted by *sort1* and the sorted version of *args2* is denoted by *sort2* in the figure.

These sorted portions are used throughout the template so that the functions can properly understand the list structures. In the above case nothing needs to be sorted. But, in general the arguments to an operator can be rearranged without any ill effects since the operator still references the correct arguments. Note that references to the original *args1* and *args2* of particular preconditions, deletions, and additions sets in the template still remain the same--thus insuring that any scrambling of arguments by the sorting leaves the effects unchanged.

The functions $P_{op(args1)}$, $D_{op(args1)}$ and $A_{op(args1)}$, return the preconditions, deletions and additions set of *op* with *args*. For example:

$$P_{putdown(u1)} = \{\text{HOLDING}(u1)\}$$
$$A_{pickup(C)} = \{\text{HOLDING}(C)\}$$

The three recursive functions **p-r***op*, **d-r***op* and **a-r***op*, defined on the second page of figure 2-3, return a set of constraints for arbitrarily long lists. For example, the function **p-r***op* returns a set which defines part of the operator's preconditions. Intuitively, the function **p-r***op* returns the preconditions of the operator with the second set of arguments (i.e. the set consists of constraints that aren't already in the preconditions of the previous operator application and those that aren't in the previous operator's add list) and then it recursively does the same thing for the next set of arguments. The functions **d-r***op* and **a-r***op* similarly return sets that represent the results of applying an operator to the members of a list in some particular fashion. The body in the template contains the tail-recursive implementation of looping. The next five steps show operator discovery and learning in the training example.

STEP 1. *Restate* the goal concept by finding the weakest preconditions for a sequence of operators such that the specific goal will still hold. From figure 2-2 the goal concept of CONSTRAINTS(C,0) can now be restated as:

$$(\exists y)\text{HOLDS}(\{\text{ON}(y,C)\},0) \wedge \text{HOLDS}(\{\text{CLEAR}(u1),\text{ON}(u1,u2),\\ \text{ON}(u2,C),\text{HANDEMPTY}\},0) \Rightarrow \text{CONSTRAINTS}(C,0)$$

Since the second conjunct in the antecedent is more specific than the first, the first conjunct can be eliminated:

$$\text{HOLDS}(\{\text{CLEAR}(u1),\text{ON}(u1,u2),\text{ON}(u2,C),\text{HANDEMPTY}\},0) \Rightarrow \text{CONSTRAINTS}(C,0)$$

[7] A Prolog-like list structure representation is used in this paper. For example: [A,B,C] is a list with three elements in it; [h|t] is a list with h as its head and t as its tail. The reason that this representation is that it avoids referring to functions such as "cdr" and "cons" or "tail" and "head", which seem to complicate the appearance of expressions occasionally.

Figure 2-3: THE GENERAL ALGORITHM THEORY OF LOOPING AND AN INSTANTIATION

The General Theory for Looping Constructs:

```
(∃op(...result(op(args2),result(op(args1),s)))

⇒

r-op([sort1|t])
    P: P_op(args1) ∪ p-r-op([sort1|t])
    D: D_op(args1) ∪ d-r-op([sort1|t])
    A: a-r-op([sort1|t])
    BODY:
            b-r-op([sort1])  <--      op(args1).
            b-r-op([comb|t]) <--      op(args1),
                                  b-r-op([sort2|t]).
```

continued on next page

The Instantiated Looping Construct

```
r-unstack-putdown([u1,u2|t])
   P: {HANDEMPTY,CLEAR(u1),ON(u1,u2)} ∪ p-r-unstack-putdown([u1,u2|t])
   D: {ON(u1,u2)} ∪ d-r-unstack-putdown([u1,u2|t])
   A: a-r-unstack-putdown([u1,u2|t])
   BODY:
      b-r-unstack-putdown([u1,u2])        <--- unstack-putdown(u1,u2).
      b-r-unstack-putdown([u1,u2,u3|t])   <--- unstack-putdown(u1,u2),
                                               b-r-unstack-putdown([u2,u3|t]).
```

continued on next page

The Existing Operator Used in the Instantiation:

```
unstack-putdown(x,y)
   P: {HANDEMPTY,CLEAR(x),ON(x,y)}
   D: {ON(x,y)}
   A: {CLEAR(y),ONTABLE(x)}
   BODY: unstack(x,y),
         putdown(x).
```

Figure 2-3, continued

The General Theory for Looping Constructs:

p-r-*op*([*sort1*]) $= \emptyset$
p-r-*op*([*comb*|t]) $= ((P_{op(args2)} - P_{op(args1)}) - A_{op(args1)}) \cup$ p-r-*op*([*sort2*|t])

d-r-*op*([*sort1*]) $= \emptyset$
d-r-*op*([*comb*|t]) $= (D_{op(args2)} - A_{op(args1)}) \cup$ d-r-*op*([*sort2*|t])

a-r-*op*([*sort1*]) $= A_{op(args1)}$
a-r-*op*([*comb*|t]) $= (A_{op(args1)} - D_{op(args2)}) \cup$ a-r-*op*([*sort2*|t])

The Instantiated Looping Construct:

p-r-unstack-putdown([u1,u2]) $= \emptyset$
p-r-unstack-putdown([u1,u2,u3|t]) = {ON(u2,u3)} $\cup$ p-r-unstack-putdown([u2,u3|t])

d-r-unstack-putdown([u1,u2]) $= \emptyset$
d-r-unstack-putdown([u1,u2,u3|t]) = {ON(u2,u3)} $\cup$ d-r-unstack-putdown([u2,u3|t])

a-r-unstack-putdown([u1,u2]) = {CLEAR(u2),ONTABLE(u1)}
a-r-unstack-putdown([u1,u2,u3|t]) = {CLEAR(u2),ONTABLE(u1)} $\cup$ a-r-unstack-putdown([u2,u3|t])

The Existing Operator Used in the Instantiation:

unstack-putdown(x,y)
P: {HANDEMPTY,CLEAR(x),ON(x,y)}
D: {ON(x,y)}
A: {CLEAR(y),ONTABLE(x)}
BODY: **unstack**(x,y),
putdown(x).

STEP 2. *Discover* algorithmic control constructs in the solution sequence. Some simple searching reveals a looping construct in the sequence of operators found in the explanation. The operator that repeats is constructed by using another algorithm theory[8] The sequence operator **unstack-putdown** is shown at the bottom of figure 2-3.

From the explanation structure. the general pattern of arguments between any two instances of **unstack-putdown** is **unstack-putdown**(u1.u2) and **unstack-putdown**(u2.u3) and that pattern holds throughout the repeating sequence. The resulting algorithm that represents **unstack-putdown** repeating an arbitrary number of times with the pattern of (u1.u2) and (u2,u3) holding throughout the sequence is instantiated as shown in the middle portion of 2-3. Below is an example of the returned set of constraints on the objects in the argument list. Note that the list can be any length.

$$P_{\text{r-unstack-putdown}((A.B.C.D))} = \{\text{HANDEMPTY},\text{CLEAR}(A),\text{ON}(A.B),\text{ON}(B,C),\text{ON}(C.D)\}$$
$$D_{\text{r-unstack-putdown}((A.B.C.D))} = \{\text{ON}(A.B),\text{ON}(B,C),\text{ON}(C.D)\}$$
$$A_{\text{r-unstack-putdown}((A.B.C.D))} = \{\text{CLEAR}(B),\text{ONTABLE}(A).\text{CLEAR}(C).\text{ONTABLE}(B), \text{CLEAR}(D),\text{ONTABLE}(C)\}$$

STEP 3. *Find* the specialization of the discovered operator's preconditions such that it implies the weakest precondition from step 1. A subscript is added to the function to denote the specialization.

$$\text{HOLDS}(P_{\text{r-unstack-putdown}(L)},0) \Rightarrow$$
$$\text{HOLDS}(\{\text{CLEAR}(u1),\text{ON}(u1.u2).\text{ON}(u2.C),\text{HANDEMPTY}\}.0) \Rightarrow \text{CONSTRAINTS}(C.0)$$

By simply matching the precondition function to the weakest precondition obtained from step 1 the following bindings occur:

$$\text{p-r-unstack-putdown}_c((u1.C)) = \phi$$
$$\text{p-r-unstack-putdown}_c((u1,u2.u3\ t)) = \{\text{ON}(u2.u3)\} \cup$$
$$\text{p-r-unstack-putdown}_c((u2,u3\ t))$$

STEPS 4 and 5. *Regress* the general goal concept through the discovered operator's preconditions and *instantiate* the discovered operator. Finally, the general goal concept of CONSTRAINTS(x,s) is regressed through the above implication and the instantiated algorithm now contains a reference to x as it's last list element in the preconditions. deletions and additions set as below and the parameter x.

[8]Not shown here. The sequence theory is not very difficult to understand. The theory references two algorithms and their arguments and combines the two into one.

```
r-unstack-putdown_x(x)
   P: {HANDEMPTY,CLEAR(u1),ON(u1,u2)} ∪ p-r-unstack-putdown_x([u1,u2|t])
   D: {ON(u1,u2)} ∪ d-r-unstack-putdown_x([u1,u2|t])
   A: a-r-unstack-putdown_x([u1,u2|t])
   BODY:
      b-r-unstack-putdown_x([u1,x])         <--- unstack-putdown_x(u1,x).
      b-r-unstack-putdown_x([u1,u2,u3|t]) <--- unstack-putdown_x(u1,u2),
                                              b-r-unstack-putdown_x([u2,u3|t]).
p-r-unstack-putdown_x([u1,x])         = ∅
p-r-unstack-putdown_x([u1,u2,u3|t])  = {ON(u2,u3)} ∪
                                        p-r-unstack-putdown_x([u2,u3|t])

d-r-unstack-putdown_x([u1,x])         = ∅
d-r-unstack-putdown_x([u1,u2,u3|t])  = {ON(u2,u3)} ∪
                                        d-r-unstack-putdown_x([u2,u3|t])

a-r-unstack-putdown_x([u1,x])         = {CLEAR(x),ONTABLE(u1)}
a-r-unstack-putdown_x([u1,u2,u3|t])  = {CLEAR(u2),ONTABLE(u1)} ∪
                                        a-r-unstack-putdown_x([u2,u3|t])
```

The problem solver now knows everything there is to know about clearing off all the blocks from a given block by invoking the operator **r-unstack-putdown**$_x$.

2.1. Discussion

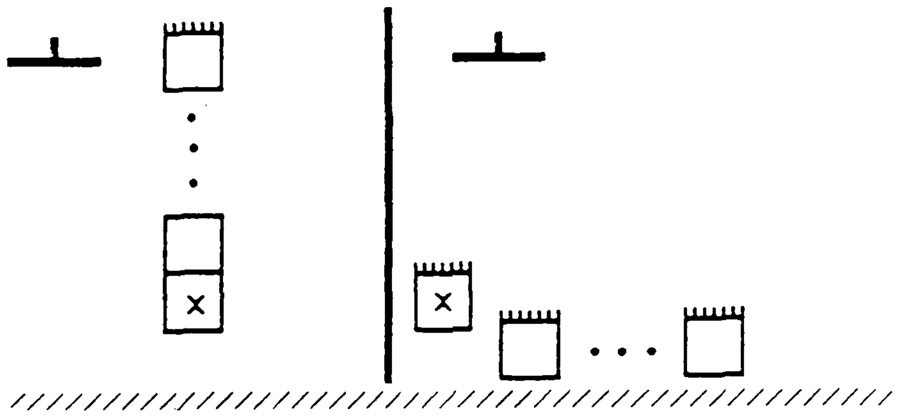
The operator that was finally learned represents a specialization or restriction of the operator originally obtained from the theory. Since the goal only specified that a certain block was to be cleared and the hand to be empty, the algorithm became tuned to this particular concept as a result of the last two steps. The operator can now clear off arbitrarily many blocks off a given block. Figure 2-4 shows the sorts of problems that it can now solve. Note that there is no reference to the fact that the given block, x, must be on the table and so that block is shown as "floating" because it doesn't matter where it is located. When the operator is applied its preconditions will result in the binding of a list of all the objects on top of x. During execution the algorithm will proceed to unstack all the objects from x and place them on the table by calling **unstack-putdown** until all the objects have been cleared from the given object and it will do so in the correct order since the list structure has preserved the sequentiality needed.

Another interesting point about this particular algorithm is that a very complex concept has been discovered in the algorithm preconditions: that of a tower with any number of blocks. Although this in itself is uninteresting from the point of view of algorithm learning, it shows how concepts with algorithmic regularity can be represented compactly and succinctly.

3. Conclusions and Summary

This section discusses some limitations of the method and some current and future research aimed at those limitations. This section concludes with a summary. As was shown from the example in section 2, the method presented in this paper is quite powerful and

Figure 2-4: PICTORIAL REPRESENTATION OF r-unstack-putdown$_x$ PRECONDITIONS AND ADDITIONS SET



general. however, *real* algorithm learning represents much more than just knowing the semantics of algorithmic constructs. Currently the limitations of this method include:

- The need for a strong theory to describe algorithmic constructs.
- Lack of knowledge about any data structures other than lists.
- Lack of knowledge about the domain theory that can be exploited in locating algorithmic constructs.
- Lack of knowledge about algorithm design (at higher levels)

Where does the theory of algorithms come from and how does the quality of the theory influence the types of algorithms that can be learned? Here the theory was derived from program verification techniques. An interesting question for future research is whether the theory itself could be derived by using another weaker theory.

The structure of the plan in explanation-based learning appears to be unimportant. In the context of learning of algorithms, however. the plan as a non-linear (dag) graph is a much better structure for purposes of discovery since independent subproblems can be located easily and the assumption of the basic argument structure of a list for algorithms constructs such as loops can be weakened to a set. Recall that the list structure was built by noting the overlapping portions or dependencies between two consecutive algorithm applications and it forces the sequentiality of the algorithm application. If, however, two consecutive algorithm applications have no common portions then a list structure is unnecessary since either one can be executed independently of the other and a set data structure is a better representation. It is in this spirit of data structure discovery that explanation-based generalization with plan graphs is being investigated.

Another interesting future topic currently being investigated is to combine domain knowledge present with operators that don't contain domain knowledge. One could imagine a naive physics theory that slowly becomes part of the operators after solving problems involving deducing some physical relations that aren't described by operators *and* using operators.[9] For example, building a tower larger than a certain height may cause towers that topple over because of unsteadiness--yet there is nothing in the operator preconditions to prevent that case from occurring even though a domain theory may express notions of balance.

There is much knowledge to be gleaned from algorithm design--that describes methods to choose subprocedures and decompose problems--that would be very useful to incorporate into the theory of algorithms.[10]

This paper presented research that extends methods used in explanation-based generalization to discovery of algorithms by combining a theory of algorithms with explanation-based generalization. The general schema for one particular algorithm was shown and used to discover an algorithmic construct that represented a generalization of the solution sequence. The discovered construct represents an enormous power increase in problem-solving performance for the planner since it embodies a theoretically infinite number of sequences compactly and succinctly. This transition from weak to strong methods involves a potentially exponential speed-up factor in problem-solving performance.

4. Acknowledgments

This paper is part of the author's thesis research. A Prolog implementation of the research presented here is currently underway. Thanks go to Tom Mitchell, Chris Tong and Jack Mostow for comments on earlier versions of this paper. Thanks also to Jaizhen Cai for many thoughtful discussions regarding set theory.

[9] This topic suggestion is due to Tom Mitchell

[10] see (Kant, 1985, Steier and Kant, 1985, Kant, 1982) on automatic programming perspectives and (Mesarovic, 1970) for information-theoretical perspectives on hierarchical systems

I. APPENDIX: Primitive Blocks-World Operators

This section list all the primitive blocks world operators. The preconditions set, P, describes what must be true in the state of the world before operator application. The deletes set, D, and the additions set, A, describe what facts will be deleted and added to the state of the world. Note that the primitive operators have no bodies since no other control constructs can be present at the primitives level.

putdown(x)
  P & D: {HOLDING(x)}
  A: {ONTABLE(x),CLEAR(x),HANDEMPTY}

pickup(x)
  P & D: {ONTABLE(x),CLEAR(x),HANDEMPTY}
  A: {HOLDING(x)}

unstack(x,y)
  P & D: {HANDEMPTY,CLEAR(x),ON(x,y)}
  A: {HOLDING(x),CLEAR(y)}

stack(x,y)
  P & D: {HOLDING(x),CLEAR(y)}
  A: {HANDEMPTY,CLEAR(x),ON(x,y)}

References

Bauer, M. *A Basis For The Aquisition of Procedures From Protocols*, pages 226-231. Proceedings IJCAI-5, Cambridge, MA. 1977.

DeJong, G. *Acquiring Schemata Through Understanding and Generalizing Plans*, pages 462-464. Proceedings IJCAI-8, Karlsruhe, West Germany, August, 1983.

Fikes, R., Hart, P., and Nilsson, N. J. Learning and Executing Generalized Robot Plans. *Artificial Intelligence*, 1972, *3*,(4), 251-288. also in *Readings in Artificial Intelligence*. Webber, B. L. and Nilsson, N. J., (Eds.).

Hoare, C.A.R. and Wirth, N. An Axiomatic Definition of the Programming Language PASCAL. *Acta Informatica*, 1973, 2,335-355.

Kant,E. and Newell,A. *Problem-Solving Techniques for the Design of Algorithms*. Technical Report CMU-CS-82-145, CMU, November 1982.

Kant, E. *Understanding and Automating Algorithm Design*. Proceedings IJCAI-9, Los Angeles, CA, August, 1985.

Korf, R. *Learning to Solve Problems by Searching for Macro-Operators*. Marshfield, MA: Pitman, 1985.

Laird, J. E., Rosenbloom, P. S., Newell, A. *Toward Chunking as a General Learning Mechanism*, pages 188-192. Proceedings AAAI-84, Austin, TX, August, 1984.

Mahadevan, S. *Verification-Based Learning: A Generalization Strategy for Inferring Problem-Decomposition Methods*. Proceedings IJCAI-9, Los Angeles, CA, August, 1985.

Manna, Z. *Mathematical Theory of Computation*. NY: McGraw-Hill 1974, 1974.

Mesarovic,M. *Theory of Hierarchical Multilevel Systems*. NY: Academic Press, 1970.

Mili, A. *An Introduction to Formal Program Verification* . NY: Van Nostrand, 1985.

Minton, S. *Constraint-Based Generalization: Learning Game-Playing Plans From Single Examples*, pages 251-254. Proceedings AAAI-84, Austin, TX, August, 1984.

Mitchell, T., Keller, R. and Kedar-Cabelli, S. Explanation-Based Generalization: A unifying view. *Machine Learning*, January 1986, *1*,(1), ?

Phillips, J. *Program Inference From Traces using Multiple Knowledge Sources*, pages 812. Proceedings IJCAI-5, Cambridge, MA, August, 1977.

Siklossy, L., Sykes, D.A. *Automatic Program Synthesis From Example Problems*, pages 268-273. Proceedings IJCAI-5, Cambridge, MA, 1977.

Steier, D. and Kant, E. *Symbolic Execution in Algorithm Design*. Proceedings IJCAI-9, Los Angeles, CA, August, 1985.

Utgoff, P. E. *Shift of Bias for Inductive Concept Learning*. PhD thesis, Department of Computer Science, Rutgers University, May, 1984.

Conceptual Clustering, Semantic Organization and Polymor

Stephen José Hanson

and

Malcolm Bauer

Bell Communications Research
435 South Street
Morristown NJ 07974

Abstract

This paper describes a machine induction program (WITT) that attempts to model human categorization. Properties of categories to which human subjects are sensitive includes best or prototypical members, relative contrasts between putative categories, and polymorphy (neither necessary or sufficient features). This approach represents an alternative to usual Artificial Intelligence approaches to generalization and conceptual clustering which tend to focus on necessary and sufficient feature rules, equivalence classes, and simple search and match schemes. We demonstrate how this categorization scheme may be used in the construction of semantic nets. Semantic variables are extracted from the 1985 World Almanac for nations of the world and used in WITT to determine clusters of nations, relationships among nations (semantic links and category structure) and underlying relations of semantic features within clusters that provide the basis of the group membership.

Introduction

Much of the current work on machine learning and conceptual clustering rests on five false premises:

(1) that necessary and sufficient features, or common feature lists, must be central to the categorization engine;

1 Parts of this paper will appear in Kanal & Lemmer, Uncertainty in Artificial Intelligence North Holland, 1986

(2) that categories are equivalence classes, which are typically treated as though there were no within category structure;

(3) that feature polymorphy (neither necessary or sufficient features) is either uninteresting or noise;

(4) that probability measures (or information measures) and symbolic manipulation are antagonistic;

(5) that the preceding four assumptions are consistent with human categorization data.

In contrast, psychological results in the categorization literature are inconsistent with each of the five assumptions above. People do not seem to try to form categories by determining the necessary and sufficient set[2] of defining features (Michalski, 1980) for a set of objects. Rather, people seem to form relative "contrasts" between categories; that is, people tend to minimize variance within clusters while maximizing variance between clusters (Rosch & Lloyd, 1978; Smith & Medin, 1981). People also tend to have best or prototypical members of a category as opposed to equivalence classes (Homa, 1978; Posner & Keele,1968); people tend to impose a category structure on a set of objects.

Many categories that people use (perhaps all natural categories) have all or at least some members that possess neither necessary nor sufficient features and can best be described by a polymorphy rule (m features out of n, $m < n$; Dennis, Hampton & Lea, 1973; Smith & Medin 1981, chapter 4, probabilistic features; see Figure 1 for an example of polymorphous categories). Finally, recent evidence suggests that the "basic level" categories, ones which people tend to reference with respect to other categories (i.e. use in natural conversation, say, in terms of something like "please sit in the chair" as compared to "please sit in the furniture") are those that maximize

2 The distinction drawn here is somewhat subtle and does not imply that people and animals do not have or know about categories that possess necessary and sufficient features, to imply that people could not use common features is contrary to intuition However, there are many possible mechanisms for achieving necessary & sufficient feature sets for categories, ones that do not require that the clustering method focus exclusively on finding necessary and sufficient feature sets.

conditional probabilities of features within a category (Rosch and Mervis, 1975). More recently, such measures have implicated the transmission of information[3] between features and categories or inter-correlations of features (cf., Murphy & Medin, 1985; Corter & Gluck, 1985).

The notion that will be put forward here is that categories that exist at some focal or basic level are also those that are bundles of feature inter-correlations which in turn tend to promote a coherence in the category (cf. Murphy & Medin, 1985). The coherence will be assumed to be related to those feature inter-correlations which promote contrast with another category. Thus, covariation that is both significant in the feature space and distinctive for clusters will be those that promote coherence in the category space which at the same time reduces the potential number of correlations that must be examined.

Rationale

In what follows we discuss a rationale for the present conceptual clustering scheme, a particular implementation and a few experiments demonstrating the unique properties of the approach. We motivate the following conceptual clustering scheme in terms of the psychological literature on categorization just indicated. This approach allows us to make clear the specific assumptions underlying the notion of conceptual clustering and the psychological nature of the properties of categories:

> *(1) Categories arise as contrasts between one another, in other words, categorization is relative to the existing context of other putative categories.*

3 Information is used in a technical sense to refer to Shannon's (1948) selective Information (H). It will be used in measures of the "likelihood" that entities belong together. Most of the measures that have been proposed actually involve information loss as contrasted with feature-feature transmission or covariation analysis of features as will be proposed here.

(2) Categories have a distribution of members, some more representative, some less. Furthermore there tends to be one best or a set of best members or an abstracted member (prototype) that may be used to represent the entire category.

(3) Categories tend to possess members that have features that are neither necessary nor sufficient (polymorphy). Polymorphy seems to arise in real world contexts, that is, when there is either natural variation in members or when the category is supported by a rational or causal account of the underlying relations between objects in the category.

(4) Categories can be represented by the inter-correlation (transmisssion) between feature sets; such inter-correlations can be used as a measure of the coherence of the concept underlying a category. The "similarity" of categories is in terms of the transmission between feature sets identified within each category.

(5) Categories and categorization should be motivated by psychological research, because categorization is a basic process that underlies many artificial intelligence domains including expert systems, natural language processing, semantic networks, as well as information retrieval. Both the human comprehensibility of the categorization and its match to human performance are crucial to the realization of progress in each of these areas.

In summary, a concept will be defined as having four properties: (1) an identity that can be described in terms of the inter-correlations in the feature space; (2) prototypical or best members; (3) layers of objects that possess fewer common and distinctive features and that introduce more polymorphy into the category: (4) a relative tension or contrast between a given concept and any other concept in the field.

The approach described in this paper is distinct from statistical clustering. which stresses descriptive models. Statistical clustering has been used primarily to provide different views of the same data or to explore data by using arbitrary similarity metrics and rules for group membership (cf. Everitt, 1977). Widely used statistical methods for clustering typically admit a few kinds of metrics (e.g. euclidian, city block) and a few kinds of group membership rules (e.g., centroid, single linkage. and complete linkage), although there are clustering packages (e.g., CLUSTAN 1B, Wishart, 1969) that effectively allow hundreds of different ways to do clustering analysis of the same data! Furthermore, statistical clustering approaches tend to focus on the entire data set in their attempt to reduce the similarity measures into a more compact representation (e.g. MDS or factor analysis).

In contrast, conceptual clustering is a process model which attempts to derive the categories that would be most consistent with semantic or structural (relations of objects) interpretation in which the members could have been described. Conceptual clustering is a "weak" approach which uses very little prior information about the nominal nature of the categories ("isa" or "kindof" or property lists); nonetheless, nothing would preclude the addition of further knowledge about the category or the input entities or the feature relations (weights or selection criteria). Basically, this approach attempts to use the known psychological properties of categories and find the most likely representation of the given entities based on input features. Finally, note that in the development of this approach feature selection is not attempted.

Because similarity can be defined in so many ways and is so controversial,[4] we assume (1) that people can form and use probability estimates within a feature space, resembling the kind of uncertainty that arises in everyday reasoning contexts (e.g., "the probability that the afternoon coffee time (which I can never remember the exact times for) is over at the cafe." (2) that these probability estimates can be used to form contrasts between potential categories (cf. Tversky, 1977); and (3) that the category structure includes prototypes, polymorphy, and a tension between generalization to other categories and the identity of a category relative to all other categories as they are forming. Finally, the present conceptual clustering approach is agglomerative and uses local views of the feature space as contrasted with a factor analytic approach or any type of divisive clustering.

4 For example, different similarity measures can be used to recover many different structures within the same data Furthermore, stimuli like words do not seem to be best described by models that are appropriate for more perceptual stimuli like colors (cf Shepard, 1978).

WITT Structure

The present conceptual clustering algorithm (WITT[5]) attempts to automatically cluster a set of objects which have been previously defined in a feature space. WITT's primary goal is to discover concepts in the object set by forming hypotheses and testing the putative concepts that result for cohesiveness. Failure of one hypothesis leads WITT to test other hypotheses that involve the creation of new concepts or the merging of old ones. Various hypotheses are attempted in sequence (an escalation over hypothesis states) in order to achieve better overall cohesiveness within all categories, and simultaneously increasing the distinctiveness between categories. At present, acceptable levels of cohesiveness are indicated to WITT through two parameters that index the relative transmission (normalized to 0,1) within and between categories.

WITT's control structure is modeled after a person attempting to sort or categorize an arbitrary number of objects into a set of disjoint, coherent categories. An example of the kinds of tasks that WITT would attempt to model might be an expert in a field of research attempting to file away some recent documents relevant to a particular sub-area of research while at the same time trying to maximize the probability of retrieving the document again in an appropriate context. Another example of a task WITT models is a problem solver attempting to decide whether the problem it is presently dealing with is similar to another problem that it has already solved in the past and subsequently deciding that it should attempt to apply the same sort of problem solving technique (these might be domain specific solutions or so-called weak methods; cf. Laird, Rosenblum, & Newell 1985). Still another kind of problem

5 Named for the philosopher Ludwig Wittgenstein who argued persuasively for "family resemblance" and polymorphy as the basis for categorization and language. His classic example of this problem is the nature of the category "game".

WITT is implemented in zeta-lisp on a Symbolics 3600.

that WITT is designed for is the addition, through analogy of new knowledge to an existing knowledge domain. WITT can determine the proximity of two concepts (e.g., based on a set of substitutive or nominal variables) relative to how that specific analogy might distinguish other existing concepts already present in the knowledge domain.

WITT has three major components: a linear goal stack, a hypothesis generator and a transmission (see below) metric to detect significant inter-relationships among features. WITT enters goal states in an attempt to find objects to add to existing categories (object-hunting), find new dense regions in the feature space (proto-seeds), or to merge together existing categories or proto-seeds.

There are only three goal states: object hunting, protoseed hunting, and protoseed merging, each with its specific procedures, although each goal state employs the same *information metric in some specific way*. WITT cycles within each state until it hits an impasse, that is, until it finds that it cannot precede with the present goal. Then the next state on the goal stack is invoked and hypotheses appropriate to the new goal state are considered.

Given a set of objects defined on a (binary or multi-valued) feature space, WITT first attempts to form local estimates of dense regions by looking for very similar objects based on an information-loss metric (cf. Orloci, 1969; Lance & Williams, 1967; Wallace & Boulton, 1968). This defines a preclustering process that gathers identical or relatively identical objects together until a threshold is reached relative to the initial pair of objects joined. Each of these regions is then assigned to a "protoseed".

The protoseed measure is a standard information-loss type measure in which the information content of the objects in terms of features when they are separate is compared with the content when they are joined. If information loss is relatively small they are subsequently assigned to a new protoseed. This measure is used first for the sake of efficiency and speed. The information-loss measure also avoids calculations of inter-correlations of a small number of objects which are likely to spurious. However, the main metric that is used for other comparisons is a transmission measure among the features with a set, between sets of objects (say, o_x and o_y) and between objects and protoseeds. Such measures are usually defined in terms of independence within the feature space as a function of an information content measure (H):

$$t(o_x;o_y) = H(o_x) + H(o_y) - H(o_x,o_y) \qquad (1)$$

and are used in WITT to establish transmission between various entities. This measure can be shown to be similar to several recent proposals of metrics for basic level categories (Hanson, in progress; Jones, 1983; Murphy, 1982; Corter & Gluck, 1985).

At each cycle WITT begins to test whether it is possible to add members to each protoseed without affecting the "identity" of each putative concept. The identity of each protoseed is determined by coherence of feature values that support and at the same time distinguish it from all other existing concepts. A ratio of the transmission to the object to a potential category (C_i) relative to all others is formed at each pass to test this tradeoff between the cohesion and distinctiveness of the category:

$$\frac{t(o_x\,;\,C_i)}{t(o_x\,;\,C_j)} > T_c \quad \text{for all} \;\; i \neq j \qquad (2)$$

This ratio measures the coherence among features in each category relative to the coherence among features to other categories presently available. If this object hunting fails, WITT tries a new hypothesis and revives the "protoseed hunting" goal state in which new dense regions are located and assigned to new protoseeds. If the identities of all present protoseeds are maintained or improved, then the protoseed is instantiated and the object hunting goal state is re-entered.

If, on the other hand, the new protoseed fails, WITT enters a new state and hypothesizes that protoseeds are too close together to yield improvement. At this point, the protoseed merging state is invoked and identities of the protoseeds are again checked. If successful, WITT returns to object hunting; otherwise, WITT gives up since further search violates input values of the requested relative tension between categories. WITT then announces that the protoseeds are well formed and indicates whether some objects are left unclustered. WITT then describes each concept in turn and the overall structure of the cluster solution.

At the end of the process several properties of the category are likely to be achieved: (1) at least one best member is identified for each category; (2) the cohesion of the category and its relative distinctiveness are reported; (3) a relative contrast between all categories is chosen to maximize identity within a category and minimize overgeneralization between categories; (4) feature relations are indicated by the correlation structure within each category; (5) and labels for the hierarchical relations of the objects are indicated at each branch of the tree including common features, distinctive features, and necessary and sufficient features (if any).

Some Results: WITT Studies

Detection of Polymorphy.

In order to demonstrate some of the properties we have been discussing, we first examine a particularly difficult artificial set of stimuli that have been used in a categorization experiment with human subjects (Dennis, Hampton Lea, 1973). An example of the artificial stimuli are shown in Figure 1 and represent a perfect example of 2 out of 3 polymorphy. In frame A of Figure 1 are 4 exemplars that can be described by the rule 2 out of 3 of the set of circle, white, and symmetric around an 90 degree origin (the number of elements in an exemplar is incidental). In contrast the 4 exemplars in frame B of this figure can be described by the rule 2 out of 3 of the set of square, black, and assymmetric around an 90 degree origin. As stated earlier a property of polymorphous categories is that the feature set that defines the inclusion rule is based on neither necessary, sufficient nor necessary and sufficient feature sets. Thus, there is no feature common to all members of the category nor is there one that is not in one category without being present in the other.

Subjects will provide similarity ratings between pairs of exemplars which seem to indicate they are sensitive to the overlapping feature sets. Subjects can also successfully categorize these exemplars into the two indicated sets in Figure 1, although they will generally not be able to state the rule they are using to do the sorting.

WITT was given these same stimuli in a binary matrix (including the incidental feature, number of components) and different clustering parameters were used in seven separate runs. Five of these seven runs resulted in unique clusterings that cover most of the significant range of the clustering parameters, which are shown in Table I. Shown in each column are the parameter values for the cohesion (Tc), distinctiveness (Td) the ratio of distinctiveness and cohesion (Tc/Td), the resultant distinctiveness ("Dis") and cohesiveness ("Coh") of the clusters, and, in the last column, the number of clusters for the solution. If one looks for both the largest cohesiveness and concurrently the smallest distinctiveness of the solution it is clear that the intermediate ratio of cohesiveness and distinctiveness (9; starred cluster value) produces the best outcome.

Tc	Td	Tc/Td	Dis	Coh	#clusters
.5	.2	2.5	.1,.1,.03	.16,.16,.5	3
.8	.2	4	.1,.1,.03	.17,.17,.35	3
.9	.1	9	.0001	.23,.23	2*
.7	.05	40	.0001	.16,.10	2
.8	.005	140	none	.10	1

Table I: Polymorphy runs

This particular solution has an extremely good distinctiveness in terms of transmission between categories (almost 0) and the largest cohesion distributed equally across the two clusters. In many ways this is an ideal cluster solution. Further note that the feature to feature inter-correlations for all eight stimuli considered together is zero. That is, the solution to this problem must involve agglomerative or local views of the data in order to find the optimal inter-correlation set. Other standard multivariate techniques would have a difficult time with such a inter-correlation matrix.

Finally in Figure 2 is the complete dendrogram of the 8 stimuli showing the cluster history as a function of transmission between features. Notice that the two centers first isolated provide the strongest possible contrast in the space, although there are other possible choices. Thereafter members are added which reduce the overall cohesion but not beneath a tolerable threshold (we actually allowed WITT to chose a relatively high value of cohesion if it could). WITT stopped because it ran out of objects to cluster. None of the exemplars were considered by WITT to be more prototypical of the discovered categories than any other.

Semantic Nets

The following demonstration is intended to show how semantic net construction might be accomplished by using machine readable resources and conceptual clustering. In this type of approach semantic relations are inferred from the the featural relations discovered during cluster formation. Following from the psychological evidence discussed earlier, we argue that semantic categories are those that are supported by a set of feature inter-correlations within each cluster. The comprehensibility and underlying rationale of each category depends on the constellation of feature values which supports one category and distinguishes it from another.

In this approach we also assume that semantic primitives, semantic links in a type hierarchy (e.g., isa and kindof links) and slot-filler structures such as scripts or frames have analogues in conceptual clustering. In conceptual clustering semantic primitives are to be found in the category structure, they are most likely to be represented by objects or meta-levels which provide the greatest support for the other members of a category. Links can be represented by those common, distinctive and polymorphous features that induce category membership. Scripts and other slot-filler structures are represented in conceptual clustering by the feature inter-relations, such covariation is analogous to "expectations" that causes a conceptual structure to be used to predict, using attribute "criteria", that an object is of a certain type rather than another.

There are various approaches to conceptual modeling that have appeared in AI. Schank and his collegues have argued for a conceptual dependency approach. This involves using some primitive concepts in a domain and re-representing more complex concepts with these primitives. So "liar", for example, might be represented as someone who transfers false knowledge from one person (themselves) to another (M-TRANS). In this approach each concept might possess expectations about information it is recieving, so for example, in the cannonical approach, if the "earthquake concept" has been activated then certain expectations about the size, the location, the extent of the destruction and the liklihood of further tremors may also be activated. In such a way scripts or frames can take advantage of possible (pre-enumerated) attribute correlations by "expecting" other features once a correlate has been observed.

The main disadvantage to semantic network construction is that these approahes tend to be ad-hoc. Many of the so-called fundamental problems in machine learning, for example the "bottleneck" problem and the "brittleness" problem arise from the non-principled construction of knowlege bases which makes the addition of new knowledge complicated and the generalization to new domains intractable.

In order to illustrate the present approach and constrast with other approaches we consider a specific example of the use of an archival distribution of semantic features for the nations of the world. We stress that we are not looking for one particular organization but rather any organziation of object types that is comprehensible and provides specific sensible hypothesis about the group membership based on feature sets. Thus, it should be noted that random organizations of nations are generally not sensible and that higher order descriptions such as "super-powers", "third world" and "poor but technologically advanced" are unlikely to arise by chance organizations alone.

The following example uses a machine readable version of the 1985 World Almanac in which three tuples of the form:

<country attribute value>

were extracted[6] from the running text in each section descibing a particular country. An excerpt of this text is shown for a particular country (FRANCE) in Figure 3. In each description there turned out to be 17 usable features for each country, these are shown in table 2, with their descriptions; they consisted of attributes like "defense budget", "religions" and "infant mortality". Thirty Seven countries were arbitrarily chosen with the constraint that they cover a a large range of continents and provide a large range of variation in featural values.

6. R.A. Amsler is responsible for the automatic extraction of the S-expresions from the machine readable text. This was accomplished by parsing both the photo-typsetting symbols and the resultant noun-phrases.

AREA	area of the country (hi,mid,lo)
LOCATION	location of the nation in the world (N-africa,indochina...)
INDUSTRIES	primary industries in the country (iron, cars, electronics..)
DEFENSE	amount spent of GNP on defense (hi,mid,lo)
CURRENCY	name of the currency of the country (dollar, riel, kyat...)
LITERACY	number in population literate (hi,mid,low)
CHIEF-CROPS	primary crops farmed by population (grains, wine, potatoes...)
MINERALS	primary minerals in the country (oil, iron, coal...)
IMPORTS	countries to which this coutry imports from (usa, france ...)
EXPORTS	countries to which this country exports to (usa, w-germany...)
TYPE	the type of government in place in the country in 1985 (republic, communist...)
LANGUAGE	the primary languages spoken by population (english, french...)
RELIGIONS	the primary religions practiced by the popuatlion (hindu, christian...)
TELEVISION-SETS	the number of tvs in the country (hi,mid,low)
NATIONAL-BUDGET	the size of the national budget as reported in 1985 (hi,mid,low)
PER-CAPITA-INCOME	the average income for a member of the population (hi,mid,low)
INFANT-MORTALITY	the rate of infant death from birth, disease etc.. (hi,mid,low)

Table 2: Features for Nations of the World

Although more nations could have been included, we felt it was not necessary since we were interested in constructing meta-level categories, which should not be necessarily based on complete coverage (i.e., you may know the concept "third world" countries but not be able to name them all). And in any case, the world almanac does not really provide comprehensive coverage over the entire world.

Such three tuples were further cleaned up and quantitative variables such as "per-cent literacy of population" were transformed to ordinal values. This was done automatically by examining the frequency histogram of the variable and looking for clusters in the data that would suggest a break for multivalued variable like "hi", "mid" or "low". Approximately half the variables were quantitative and transformed accordingly. Nonetheless, WITT, only uses the nominal or semantic value of the variable without note of possible order information, although, the value is reported along with feature correlations.

WITT was given the 37 nations of the world with their corresponding attribute values. Several runs were attempted, the most stable run, that is, one that did not either find[7] one group or a large number of two country groups is shown in Figure 4.

This Figure shows the complete dendrogram for the 37 Nations of the world. Examination of the dendrogram reveals that WITT did discover reasonable and comprehensible groupings in the data. For example, at the top of the dendrogram note the first cluster of 3 countries including USA, CANADA, and JAPAN, this set seems to be broken off from a european cluster including ITALY, SPAIN, UNITED KINGDOM, and FRANCE, which in turn are distinct from a larger cluster starting near the bottom of the dendrogram with CAMBODIA, VIETNAM and THAILAND. Toward the center of the dendrogram we have some interesting clusters such as IRELAND and ISRAEL and a separate cluster involving CHINA and USSR. Roughly the structure of the dendrogram can be broken into 7 or 8 groups. At the highest level we see a split between countries that may be described as "third world" and countries that are technologicaly advanced and have a relatively high quality of life. Finer distinctions can be made as we move down the dendrogram and notice a cluster having to do with european countries and another cluster down the tree in south-east asia or africa. However, geography seems to have less to do with the finer groupings than does some abstract qualities having to do with economy, quality of life and industries. We turn next to a specific look at some selected groupings.

In Figure 5 we show four cluster groups at the lowest level of the tree. These include the USA cluster, the european cluster the south-east asia cluster and the middle cluster including IRELAND, ISRAEL, BELGIUM and DENMARK. In each box we include attribute correlations that were amoung the top 5-10 in characterizing the cluster. For example, in the USA group we see that "high quality of life" seems to predominate. This seems indicated by the large number of television sets and the

7. We should note there were other runs, all however, bore a similarity to the groupings in Figure 4. In fact, from experience we have found there are typically only a few number of clusters that WITT can find and they are generally in a small neighborhood of possible groupings. This partly must reflect WITT's conservative control structure and partly must reflect the given distribution of attribute values in a given data set.

large national budgets each of these countries have. Also location in the world and industries seem to be a good predictor of this group, although, one that might be expected to change when other locations outside of north-america were found to contain similar values. Finally, a third predictor set was the predominate literacy of the population and the type of government which was democratic. Each of these sets were above .6 in correlation and indicate what is considered "important" to these clusters. In the next group which includes CAMBODIA, we see that "low quality of life" seems to hold the forefront, however economy seems also to be an important attribute of the cluster. For example, the pair low national budget and low per-capita income seem to be included in a number of correlations with other variables like literacy, defence and and number of televisions, a seeming mix of economic, governmental, educational and technology variables. A third set of variable correlations seem significant that involve infant mortality with location, exports and language. Such covariates would seem to implicate not only quality of life issues but hospital care, local health conditions and education.

In between these two extremes in technology and quality of life we see two other clusters shown in the bottom of figure 5 that represent intermediate values of variables that played an important role in the first two clusters. For example, the ITALY, SPAIN cluster is characterised by mid level of per-capita income and low infant mortality, in fact the lo-infant mortality feature picks up medium number of tvs and government type as well as hi literacy. Overall, we might characterize the quality of life as mid to high with main emphasis on variables like education, personal economics health care and government, contrasting with variables like national budget, defense or high technology. Finally, another intermediate case is the IRELAND, ISRAEL cluster which seems focused on attribute correlations such as hi per-capita income and low infant mortality, low number of televisions with hi to mid per-capita income. So unlike the ITALY, SPAIN cluster here we have a contrast between a relatively high quality of life but a low number of televisions and relatively poor governments which may be indicative of the local and global resources being diverted to activities other than recreation or entertainment. It may also be related to countries that are relatively isolated from the more global communities for any number of reasons including conflicts within and outside the country as well as just an intended isolationism attitude by the country itself.

Further experiments testing the comprehensiblity of these categories to people should help confirm and validate the approach. Specifically, we would expect that both the clusters of countries and the feature relations within countries to be more comprehensible to people than random organizations of nations and random correlates of features shown to subjects. Sortings of these countries should also produce similar clusterings amoung subjects asked to sort for similarity. We think that this demonstration provides both reasonable clusters of nations, ones that most people would think of as well as those that might be more subtle at first glance but in close examination provide plausible accounts of nation's relationships. Without such conceptual clustering it is easy to miss important and useful links within the semantic structure, and more critically it makes it easy to add new nodes with the automatic adjustment of the network to accomodate this new information.

Acknowledgment

We are grateful to R. B. Allen, G. W. Furnas, G. A. Miller and D. Walker for reading and commenting on an earlier version this paper.

REFERENCES

1. Butler, K. A. and Corter J. E. Use of psychometric tools for knowledge acquisition : A case study. *paper presented at Workshop on artificial Intelligence in statistics,* (1985).
2. Clancy, W. J. Classification Problem Solving *paper presented at AAAI,* (1984) pp. 49-55.
3. Corter J. M. and Gluck M. A. Machine generalization and human categorization: an information-theoretic view, *Workshop on Uncertainty and Probability in Artificial Intelligence,* UCLA, August, 1985.
4. Dennis I., Hampton J. A., Lea S. E. G., New Problem in concept formation *Nature* **243** (1973) pp. 101-102.
5. Everitt, B., *Cluster Analysis* (Heinemann Educational Books, London, 1977).
6. Homa, D., Abstraction of ill-defined form, *Journal of Experimental Psychology: Human Learning and Memory* **4** (1978) 407-416.
7. Jones G. V., Identifying basic categories, *Psychological Bulletin,* **94** (1983) 423-428.
8. Laird, J. E., Rosenbloom, P. S. and Newell, A. Towards chunking as a general learning mechanism *Technical Report cmu-cs-85-110, Department of Computer Science, CMU, January, 1985.*
9. Lance, G. N., Williams, W. T. Note on a new information-statistic classificatory program *Computer Journal* (1967) 195.
10. Michalski, R. S., Knowledge acquisition through conceptual clustering: A theoretical framework and an algorithm for partitioning data into conjunctive concepts, *International Journal of Policy Analysis and Information Systems* **4** (1980) 219-244.
11. Michalski, R. S. and Stepp, R. E., Learning from observation: conceptual clustering. in: R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Tioga, Palo Alto, 1983) 331-363.
12. Murphy, G. L., Cue validity and levels of categorization, *Psychological Bulletin,* **91** (1982) 174-177.
13. Murphy, G. L. and Medin, D. L. The role of theories in conceptual coherence, *Psychological Review,* **92** (1985) 289-316.
14. Orloci, L., Information analysis of structure in biological collections, *Nature* **223** (1969) 483-484.
15. Posner, M. I. and Keele, S. W., On the genesis of abstract ideas, *Journal of Experimental Psychology* **77** (1968) 353-363.
16. Reed, S. K., Pattern recognition and categorization, *Cognitive Psychology* **3** (1972) 382-407.
17. Rosch, E. and Lloyd, B. B. (Eds.), *Cognition and Categorization* (Erlbaum, Hillsdale, NJ, 1978).
18. Sejnowski, T. J., Kienker, P. K. and Hinton, G. E. Learning Symmetry groups with hidden units: Beyond the perceptron, *Submitted to Physica D (1986).*
19. Shannon, C. E., A mathematical theory of communication, *Bell System Technical Journal* **27** (1948) 379-423, 623-656.

20. Shepard R. N., Multidimensional Scaling, Tree-Fitting and Clustering, *Science* **210** (1980) 390-398.

21. Smith, E. E. and Medin, D. L., *Categories and Concepts* (Harvard Univ. Press, Cambridge, MA, 1981).

22. Wallace, C. S. and Boulton D. M., An information measure for classification. *Computer Journal.* **11** (1968) 185-194.

23. Wishart, D., Numerical classification method for deriving natural classes, *Nature* **221** (1969) 97-98.

24. Tversky, A. Features of Similarity *Psychological Review* **84** (1977) 327-352.

Figure Legends

Figure 1: Two out of three polymorphy example adapted from Dennis, Hampton and Lea, 1973.

Figure 2: Complete dendrogram for the polymorphy example.

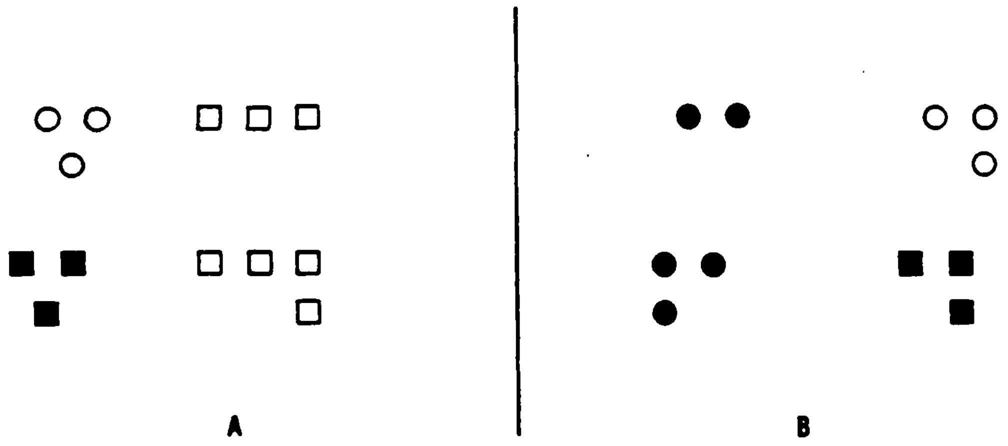
Figure 3: Excerpt from 1985 World Almanac for the nation: FRANCE

Figure 4: Dendrogram for Nations of World

Figure 5: A more Detailed view of a few meta-level concepts in the Clustering

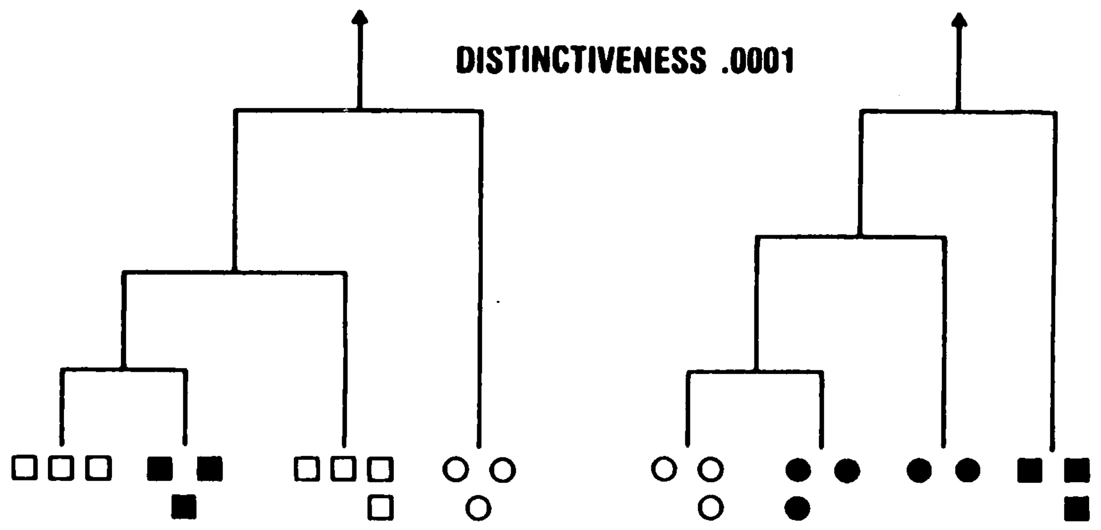
Figure 1

TWO OUT OF THREE POLYMORPHY



A: AT LEAST 2 OUT OF THE 3 FEATURES - SQUARE, WHITE, OR SYMMETRIC

B: AT LEAST 2 OUT OF THE 3 FEATURES - CIRCLE, BLACK, OR ASYMMETRIC



SQUARE & SYMMETRIC

TRANSMISSION WITHIN .23

FEATURE CORRELATIONS:

SHAPE-COLOR	.082
COLOR-ORIENT	.082
ORIENT-SHAPE	.082
ORIENT-NUMBER	1.0

NO BEST EXEMPLAR

CIRCLE & ASSYMMETRIC

TRANSMISSION WITHIN .23

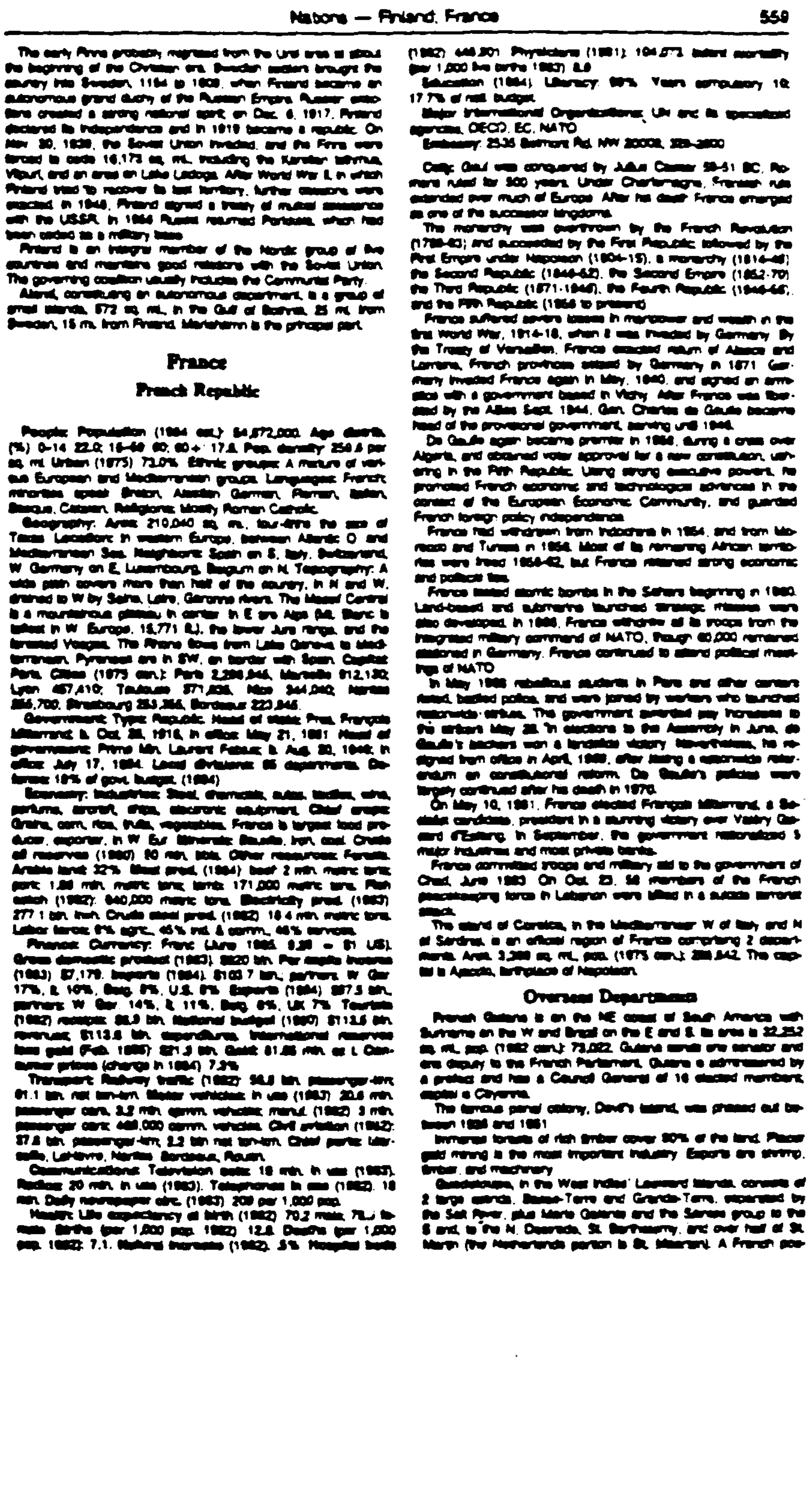
FEATURE CORRELATIONS:

SHAPE-COLOR	.082
COLOR-ORIENT	.082
ORIENT-SHAPE	.082
ORIENT-NUMBER	1.0

NO BEST EXEMPLAR

Figure 2

Figure 3



The early Finns probably migrated from the Ural area at about the beginning of the Christian era. Swedish settlers brought the country into Sweden, 1154 to 1809, when Finland became an autonomous grand duchy of the Russian Empire. Russian exactions created a strong national spirit; on Dec. 6, 1917, Finland declared its independence and in 1919 became a republic. On Nov. 30, 1939, the Soviet Union invaded, and the Finns were forced to cede 16,173 sq. mi., including the Karelian Isthmus, Viipuri, and an area on Lake Ladoga. After World War II, in which Finland tried to recover its lost territory, further cessions were exacted. In 1948, Finland signed a treaty of mutual assistance with the USSR. In 1956 Russia returned Porkkala, which had been ceded as a military base.

Finland is an integral member of the Nordic group of five countries and maintains good relations with the Soviet Union. The governing coalition usually includes the Communist Party.

Aland, constituting an autonomous department, is a group of small islands, 572 sq. mi., in the Gulf of Bothnia, 25 mi. from Sweden, 15 mi. from Finland. Mariehamn is the principal port.

France

French Republic

People: Population (1984 est.): 54,872,000. Age distrib. (%): 0-14: 22.0; 15-59: 60; 60+: 17.8. Pop. density: 250.8 per sq. mi. Urban (1975): 73.0%. Ethnic groups: A mixture of various European and Mediterranean groups. Languages: French; minorities speak Breton, Alsatian German, Flemish, Italian, Basque, Catalan. Religions: Mostly Roman Catholic.

Geography: Area: 210,040 sq. mi., four-fifths the size of Texas. Location: In western Europe, between Atlantic O. and Mediterranean Sea. Neighbors: Spain on S, Italy, Switzerland, W. Germany on E, Luxembourg, Belgium on N. Topography: A wide plain covers more than half of the country, in N and W, drained to W by Seine, Loire, Garonne rivers. The Massif Central is a mountainous plateau in center. In E are Alps (Mt. Blanc is tallest in W. Europe, 15,771 ft.), the lower Jura range, and the forested Vosges. The Rhone flows from Lake Geneva to Mediterranean. Pyrenees are in SW, on border with Spain. Capital: Paris. Cities (1975 cen.): Paris 2,290,845; Marseille 912,130; Lyon 457,410; Toulouse 371,835; Nice 344,040; Nantes 255,700; Strasbourg 253,355; Bordeaux 223,845.

Government: Type: Republic. Head of state: Pres. François Mitterrand; b. Oct. 26, 1916; in office: May 21, 1981. Head of government: Prime Min. Laurent Fabius; b. Aug. 20, 1946; in office: July 17, 1984. Local divisions: 95 departments. Defense: 18% of govt. budget. (1984)

Economy: Industries: Steel, chemicals, autos, textiles, wine, perfume, aircraft, ships, electronic equipment. Chief crops: Grains, corn, rice, fruits, vegetables. France is largest food producer, exporter, in W. Eur. Minerals: Bauxite, iron, coal. Crude oil reserves (1980): 50 mln. bbls. Other resources: Forests. Arable land: 32%. Meat prod. (1984): beef: 2 mln. metric tons; pork: 1.80 mln. metric tons; lamb: 171,000 metric tons. Fish catch (1982): 640,000 metric tons. Electricity prod. (1983): 277.1 bln. kwh. Crude steel prod. (1982): 18.4 mln. metric tons. Labor force: 9% agric., 45% ind. & comm., 46% services.

Finance: Currency: Franc (June 1985: 9.28 = $1 US). Gross domestic product (1983): $520 bln. Per capita income (1983): $7,179. Imports (1984): $103.7 bln.; partners: W. Ger. 17%, It. 10%, Belg. 9%, U.S. 8%. Exports (1984): $97.5 bln.; partners: W. Ger. 14%, It. 11%, Belg. 9%, UK 7%. Tourists (1982) receipts: $8.3 bln. National budget (1980): $113.6 bln. revenues; $113.6 bln. expenditures. International reserves less gold (Feb. 1985): $21.3 bln. Gold: 81.85 mln. oz t. Consumer prices (change in 1984): 7.3%.

Transport: Railway traffic (1982): 56.8 bln. passenger-km; 61.1 bln. net ton-km. Motor vehicles: in use (1983): 20.6 mln. passenger cars, 3.2 mln. comm. vehicles; manuf. (1982): 3 mln. passenger cars; 468,000 comm. vehicles. Civil aviation (1982): 37.8 bln. passenger-km; 2.2 bln net ton-km. Chief ports: Marseille, LeHavre, Nantes, Bordeaux, Rouen.

Communications: Television sets: 18 mln. in use (1983). Radios: 20 mln. in use (1983). Telephones in use (1982): 18 mln. Daily newspaper circ. (1983): 209 per 1,000 pop.

Health: Life expectancy at birth (1982): 70.2 male; 78.4 female. Births (per 1,000 pop. 1982): 12.8. Deaths (per 1,000 pop. 1982): 7.1. Natural increase (1982): .5%. Hospital beds (1982): 646,201. Physicians (1981): 104,573. Infant mortality (per 1,000 live births 1983): 8.9.

Education (1984): Literacy: 99%. Years compulsory: 10; 17.7% of natl. budget.

Major International Organizations: UN and its specialized agencies, OECD, EC, NATO.

Embassy: 2535 Belmont Rd. NW 20008; 328-2600.

Celtic Gaul was conquered by Julius Caesar 58-51 BC; Romans ruled for 500 years. Under Charlemagne, Frankish rule extended over much of Europe. After his death France emerged as one of the successor kingdoms.

The monarchy was overthrown by the French Revolution (1789-93) and succeeded by the First Republic; followed by the First Empire under Napoleon (1804-15), a monarchy (1814-48), the Second Republic (1848-52), the Second Empire (1852-70), the Third Republic (1871-1946), the Fourth Republic (1946-58), and the Fifth Republic (1958 to present).

France suffered severe losses in manpower and wealth in the first World War, 1914-18, when it was invaded by Germany. By the Treaty of Versailles, France exacted return of Alsace and Lorraine, French provinces seized by Germany in 1871. Germany invaded France again in May, 1940, and signed an armistice with a government based in Vichy. After France was liberated by the Allies Sept. 1944, Gen. Charles de Gaulle became head of the provisional government, serving until 1946.

De Gaulle again became premier in 1958, during a crisis over Algeria, and obtained voter approval for a new constitution, ushering in the Fifth Republic. Using strong executive powers, he promoted French economic and technological advances in the context of the European Economic Community, and guarded French foreign policy independence.

France had withdrawn from Indochina in 1954, and from Morocco and Tunisia in 1956. Most of its remaining African territories were freed 1958-62, but France retained strong economic and political ties.

France tested atomic bombs in the Sahara beginning in 1960. Land-based and submarine launched strategic missiles were also developed. In 1966, France withdrew all its troops from the integrated military command of NATO, though 60,000 remained stationed in Germany. France continued to attend political meetings of NATO.

In May 1968 rebellious students in Paris and other centers rioted, battled police, and were joined by workers who launched nationwide strikes. The government granted pay increases to the strikers May 26. In elections to the Assembly in June, de Gaulle's backers won a landslide victory. Nevertheless, he resigned from office in April, 1969, after losing a nationwide referendum on constitutional reform. De Gaulle's policies were largely continued after his death in 1970.

On May 10, 1981, France elected François Mitterrand, a Socialist candidate, president in a stunning victory over Valéry Giscard d'Estaing. In September, the government nationalized 5 major industries and most private banks.

France committed troops and military aid to the government of Chad, June 1983. On Oct. 23, 58 members of the French peacekeeping force in Lebanon were killed in a suicide terrorist attack.

The island of Corsica, in the Mediterranean W of Italy and N of Sardinia, is an official region of France comprising 2 departments. Area: 3,369 sq. mi.; pop. (1975 cen.): 289,842. The capital is Ajaccio, birthplace of Napoleon.

Overseas Departments

French Guiana is on the NE coast of South America with Suriname on the W and Brazil on the E and S. Its area is 32,252 sq. mi.; pop. (1982 cen.): 73,022. Guiana sends one senator and one deputy to the French Parliament. Guiana is administered by a prefect and has a Council General of 16 elected members; capital is Cayenne.

The famous penal colony, Devil's Island, was phased out between 1938 and 1951.

Immense forests of rich timber cover 90% of the land. Placer gold mining is the most important industry. Exports are shrimp, timber, and machinery.

Guadeloupe, in the West Indies' Leeward Islands, consists of 2 large islands, Basse-Terre and Grande-Terre, separated by the Salt River, plus Marie Galante and the Saintes group to the S and, to the N, Desirade, St. Barthelemy, and over half of St. Martin (the Netherlands portion is St. Maarten). A French pos-

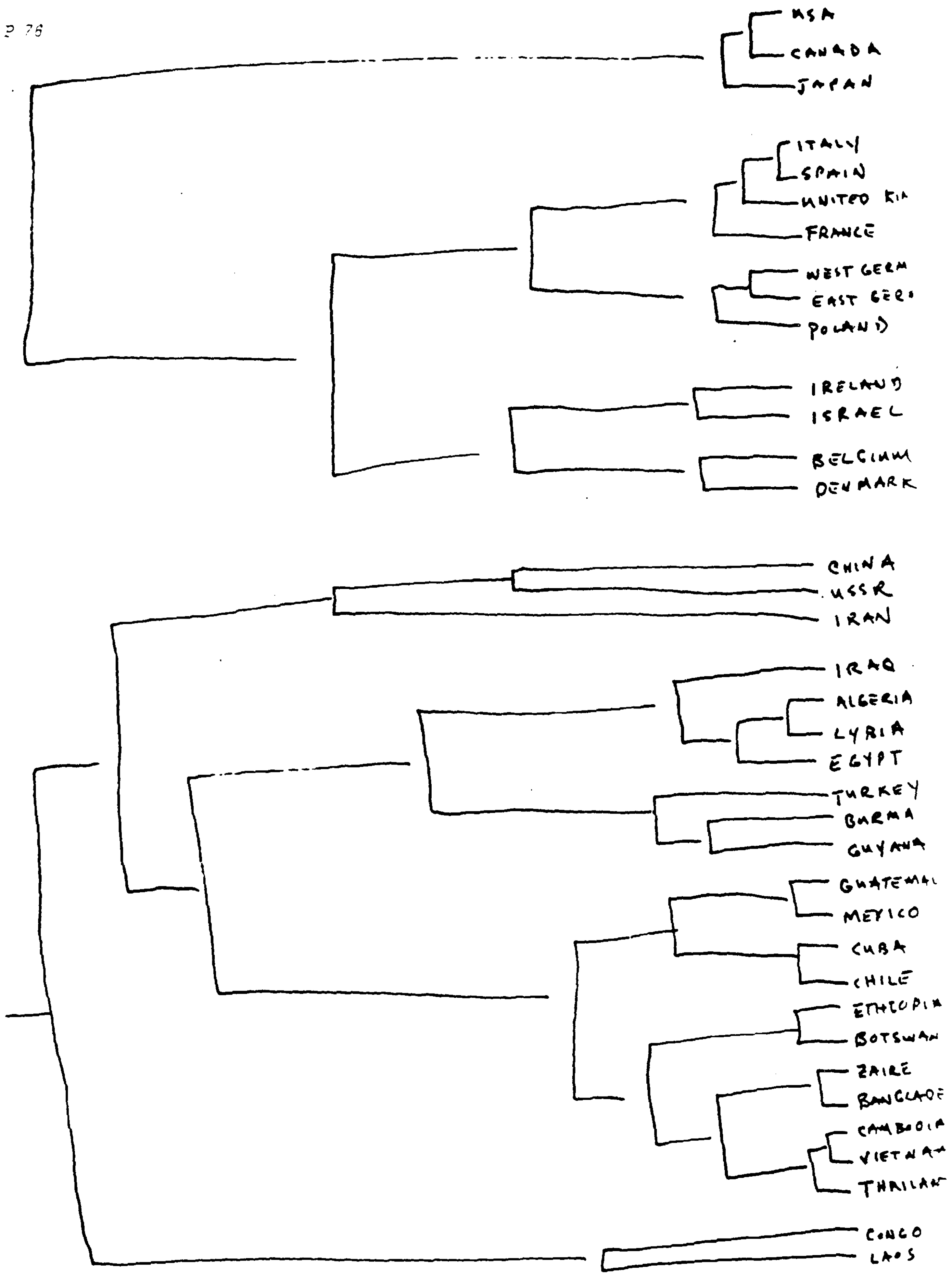

USA
CANADA
JAPAN
ITALY
SPAIN
UNITED KIN
FRANCE
WEST GERM
EAST GERM
POLAND
IRELAND
ISRAEL
BELGIUM
DENMARK
CHINA
USSR
IRAN
IRAQ
ALGERIA
LYBIA
EGYPT
TURKEY
BURMA
GUYANA
GUATEMAL
MEXICO
CUBA
CHILE
ETHIOPIA
BOTSWAN
ZAIRE
BANGLADE
CAMBODIA
VIETNAM
THAILAN
CONGO
LAOS

Figure 5

Selected Group Memberships	
Group 1	Group 2
USA Canada Japan TV (hi) x National Budget (hi) Location (north-america) x Industries (cars steel) Literacy (hi) x type (democracy)	Cambodia Vietnam Thailand National Budget (lo) x Per Capita Income (lo) TV(lo) x Per Capita Income (lo) TV(lo) x national budget(lo) Literacy (lo) x TV (lo) Defence(lo) x National Budget(lo) Exports x infant mortality(hi) Language x infant mortality(hi)
Group 3	Group 4
Italy Spain France United Kingdom Per-Capita Income (mid) x Infant Mortality (lo) TVs (mid) x Infant Mortality (lo) Type x Infant Mortality (lo) Literacy(hi) x Infant Mortality (lo)	Ireland Israel Belgium Denmark Per-Capita Income (mid-hi) x Infant Mortality(lo) National-Budget x Infant Mortality (lo) TVs(lo) x Per-Capita Income (Hi-mid) Literacy (hi-mid) x Type

THE USE OF DOMAIN PROPERTIES EXPRESSED AS THEOREMS IN MACHINE LEARNING

Christel Vrain
Laboratoire de Recherche en Informatique - UA 410 du CNRS
Université de Paris Sud
91405 ORSAY CEDEX
(1) 69-41-62-85

Topic: Learning

Keywords: generalization, axiom, structural matching

Abstract:
In this paper, we propose a generalization algorithm based on Structural Matching. At each step, we choose a constant in each example, we replace *all* its occurences with a generalization variable and we try to erase the discriminating occurences.
It is the third operation which interests us: how to make discriminating occurences disappear. We define precisely what a discriminating occurence is, we show how all kinds of axioms must be used and we define classes of predicates to decrease the search.

1. Introduction

1.1. Similarity Based Learning

In the field of Learning, there now exist two kinds of learning: Explanation Based Learning (Dejong 86, Mitchell 86) and Similarity Based Learning (Michalski 84).

The method, in Explanation Based Learning, can be divided into two steps:
- first of, we try to solve a problem like, for instance in the LEX system developed by T. Mitchell and Als. (Mitchell 83), the integration of a function,
- we then try to explain each step of the resolution in order to generalize this type of resolution, or the cases to which it can be applied.

In Similarity Based Learning, we have examples of objects, such as cancerous cells or diseases symptoms (Michalski 84) and we generalize them according to the similarities in their descriptions.

In this paper, we are concerned with this second kind of learning: we have examples $E_1, \ldots, E_n$ and we want to learn a generalization of these examples.

1.2. Structural Matching

The method used to detect similarities between examples, in this paper, has been called Structural Matching. The basic idea is to transform the representation of the examples, using axioms, properties, ..., until there is no need to use the dropping condition rule to generalize them.

For instance, consider the two following examples:

E_1 E_2

They can be represented by the formulae:
RE_1 = (square A)
RE_2 = (rectangle B)
RE_1 and RE_2 are not similar because in RE_1 there is the predicate "square" and in RE_2, there is the predicate "rectangle".
Suppose now that we know the rule:
$\forall x$ *[(square x) $\Rightarrow$ (rectangle x)]*
which expresses that a square is a kind of rectangle, we can transform RE_1 into RE_1':
RE_1' = (rectangle A)
RE_1' and RE_2 are now similar: if we consider the formula F = (rectangle x) and the substitutions σ_1, defined by $\sigma_1(x) = A$ and σ_2, defined by $\sigma_2(x) = B$, $\sigma_1(F) = RE_1$ and $\sigma_2(F) = RE_2$. We say that RE_1' and RE_2 match structurally. We now know that the objects in E_1 and in E_2 are rectangles and that a generalization G of these two examples is:

$$G = (rectangle\ x)$$

Definition 0

Let $E_1, E_2, \ldots, E_n$ be n examples.
We say that $E_1, E_2, \ldots, E_n$ **match structurally** if there exist a formula F and n substitutions σ_i such that for each i, $\sigma_i(F) = E_i$.

In our previous example, RE_1 and RE_2 did not match structurally. However, RE'_1 and RE_2 match structurally.

When examples match structurally, it is then easy to find a generalization of them: the formula F is already a kind of generalization. We can find a better generalization by analysing the substitutions σ_i.
For instance, if we consider the two following examples:
E_1 = (book A) & (red A) & (on A TABLE) & (pen B) & (on B TABLE)
E_2 = (book C) & (red C) & (on C TABLE) & (pen D) & (on D C)
These two examples match structurally, for if we consider the formula G defined by:
G = (book x) & (red x) & (on x u) & (pen y) & (on y v)
and the substitutions σ_1 defined by $\sigma_1(x) = A$, $\sigma_1(y) = B$, $\sigma_1(u) = TABLE$, $\sigma_1(v) = TABLE$ and σ_2 defined by $\sigma_2(x) = C$, $\sigma_2(y) = D$, $\sigma_2(u) = TABLE$, $\sigma_2(v) = C$, we have $\sigma_1(G) = E_1$ and $\sigma_2(G) = E_2$.
We notice that in σ_1 and in σ_2, the variable u is instantiated by the same constant TABLE. A better generalization G' of E_1 and E_2 is therefore:
G' = G & (= u TABLE).
We can further refine our generalization by comparing the instantiations of the variables in each substitution. In σ_1 and in σ_2, the variables x and y are instantiated by two different constants, A and B in σ_1 and C and D in σ_2. We can generalize this by adding to G' the fact that x and y are different. It is also true for x and u, y and u and y and v. It is not true for x and v, because, in σ_2, x and v are instantiated by the same constant C, and it is not true for u and v.
A still better generalization of G is:
G' = G & (= u TABLE) & (≠ x y) & (≠ x u) & (≠ y v) & (≠ y u).

To find this better generalization, we compared the instantiations of the variables in each substitution with the relation "=". We could do the same with other relations. For instance, if the instantiations of the variables x and y in the examples were integers, we could compare them with the relation "<" or we could compute the values of x/y in each example to see if it were constant or not, ...

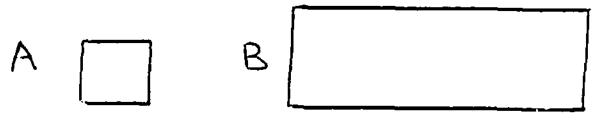
There is no end to the process of generalization refinement [Michalski 1984], but this paper is rather devoted to the study of the task to put the examples in Structural Matching. To do this, two kinds of axioms may be used:
– logic axioms like for instance, the idempotency of the conjunction: A $\Longleftrightarrow$ A & A.
– properties of the universe in which we are working. For instance, if we want to generalize pictures of geometric objects, we may use the property that a square is a kind of rectangle.

2. AGAPE

A generalization algorithm AGAPE has been implemented in our group, by T. Bollinger (Bollinger 86). It is based on the principle of Structural Matching. To put the examples in Structural Matching, idempotency is used, as well as some specific properties of the universe (those which express relations of generality between objects). In this paper, we show how all kinds of axioms must be used to put the examples into Structural Matching and we define classes of predicates to improve the algorithm of Structural Matching.

2.1. Representation of the examples

We work in first order logic. Depending upon what we want to learn, we can represent examples as conjunctions of atoms or as rules.
For instance, the following example:

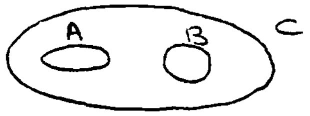


may be represented by the conjunction of atoms:
(square A) & (small A) & (rectangle B) & (large B)
Others means of representation may be used, we just have to specify how the Structural Matching must be carried out.
For instance, we can represent ~~examples such~~ as:

[CNTXT] LM -> RM

where CNTXT represents the context in which the rule can be applied. It was used to generalize Vere 's examples concerning the different ways of moving cubes (Vere 80): CNTXT described the position of the fixed cubes, LM the position of the mobile cubes before they were moved and RM their position afterwards.
With this kind of representation and for this kind of examples, we had to put the contexts into Structural Matching on the one hand, and the left handsides, and then the right handsides on the other hand. In other examples, contexts only can be generalized, while we cannot generalize the "actions".

There are two restrictions to the use of the first order logic:
– we do not allow the presence of terms, but only of constants in the representation of the examples. Sometimes, because of this restriction, some characteristics of the objects in the examples cannot be described.
For instance, suppose that we want to represent the following example:



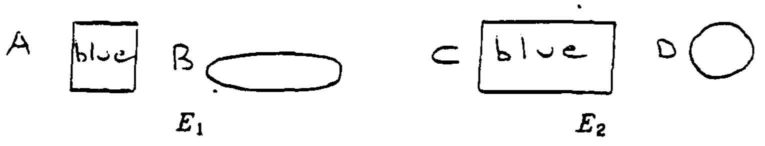
we can express that A is included in C by the atom (included A C), we can also express that B is included in C but we cannot express that A ∪ B is also included in C, because A ∪ B is not a constant.
We should allow functions in the representation of the examples and our last example could be represented by:
(included A C) & (included B C) & (included (union A B) C)
– nor do we allow negations.

In this paper, and only to simplify our presentation, we shall moreover suppose that there are no predicates of arity 0. We can easily extend our algorithm to this case.

In the next section, we give an example of the generalization algorithm and then we describe the main steps of this algorithm.

2.2. **Example**

Suppose that we have the two following examples:



which can be described by:
E_1 = (square A) & (blue A) & (ellipsoid B)
E_2 = (rectangle C) & (blue C) & (circle D)

In each example, there are two constants. Suppose that we choose the constants A and C because the objects represented by these constants are blue, and because we know that a square is a kind of rectangle. We replace all the occurences of the constant A in E_1, and all the occurences of the constant C in E_2 by the pseudo-variable x. This is not a true variable, since its value is kept in the formula in brackets. This gives the substitution that should be applied to find the original example again. But, it will perhaps become a generalization variable during the second step of the process. Therefore, we shall call it in the following, a VGE, (from the French "Variable de Généralisation Eventuelle"), which means a Tentative Generalization Variable.

E_1 = (square x) & (blue x) & (ellipsoid B) [(= x A)]
E_2 = (rectangle x) & (blue x) & (circle D) [(= x C)]

In this paper, we are not interested in the information kept in the brackets, it has already been treated by J.G. Ganascia (Ganascia 85). So, we only take into account the occurences of x, which appear in the conjunction of atoms.

In E_1, we consider the two occurences of the variable x defined: as argument of the predicate "square", and as argument of the predicate "blue".
In E_2, we consider the two occurences of the variable x defined: as argument of the predicate "rectangle" and as argument of the predicate "blue".
In E_1, the occurence of x in the atom (square x) discriminates E_1 in E_2, because we do not find it in E_2. In the following, we shall name such an occurence a **discriminating occurence**. In E_2, the occurence of x in the atom (rectangle x) is also discriminating. If we know no axioms, we must stop because we cannot make these discriminating occurences disappear and therefore we cannot put the examples into Structural Matching. But, if we know that a square is a rectangle, we can transform E_1 into E_1':
E_1' = (rectangle x) & (blue x) & (ellipsoid B) [(= x A)]
There are now no discriminating occurences of x in E_2 and E_1'.

We go on. We now choose the last constants C and D and replace them by the variable y.
E_1' = (rectangle x) & (blue x) & (ellipsoid y) [(= x A) (= y B) (≠ x y)]
E_2 = (rectangle x) & (blue x) & (circle y) [(= x C) (= y D) (≠ x y)]

We have added, for each example, the link ($\neq$ x y) in brackets, because the values of x and y are different.
In E_1', there is only an occurence of y in the atom (ellipsoid y). It is a discriminating one, because we do not find such an occurence in E_2.
In E_2, there is only an occurence of y in the atom (circle y). It is also a discriminating one.
If we know the rule: $\forall$ x [(circle x) $\Rightarrow$ (ellipsoid x)],
we can transform E_2 into E_2':
E_2' = (rectangle x) & (blue x) & (ellipsoid y) [(= x C) (= y D) ($\neq$ x y)]
The occurences of y in E_1' and in E_2' match structurally.
There are no more constants to be chosen.

We have transformed E_1 into E_1' and E_2 into E_2' so that E_1' and E_2' match structurally. Moreover, we know that in E_1 and in E_2, the instantiations of x and y are different. A generalization of E_1 and E_2 is therefore:
G = (rectangle x) & (blue x) & (ellipsoid y) & ($\neq$ x y)
which means that in E_1 and in E_2, there are two different objects: a blue rectangle and an ellipsoid.

2.3. Algorithm

Suppose we have n examples E_1, E_2, ..., E_n that we want to generalize. Our generalization method consists of two steps:
First step: we put the examples into Structural Matching.
Second step: when the examples match structurally, we can then generalize them. There exist a formula F and n substitutions σ_i so that for each i, $\sigma_i(F) = E_i$. A generalization of the examples is made from the common formula F and the common links between the variables.

In this paper, we are interested in the first step: how to put the examples into Structural Matching. The algorithm used is quite simple: we repeat the following operations until the examples match structurally, or until we have to stop.

1- At each step, according to some heuristics, we choose a constant in each example.

2- We replace all the occurences of these constants by a same Tentative Generalization Variable, that we call a VGE.

3- We compare the occurences of this VGE in the examples and we try to make all the discriminating occurences disappear. The notion of discriminating occurence will be defined in section 3.1..

4- If it remains constants which have not been already chosen, we effect the same operations 1, 2 and 3 with new constants, otherwise we stop.

This algorithm is a little simplified. In fact, during the algorithm, we may use axioms like:
$\forall$ x [(A x) $\Leftrightarrow$ (A x) & (A y) & (= x y)], A being a given predicate.
Such axioms introduce new variables, these variables are not VGEs and are treated as constants. These axioms are used in very specific cases, precisely defined and controled and we shall not speak about them in this paper.

2.4. Importance of the axioms

In this paragraph, we only want to show how all kinds of axioms must be used. For this, consider the two following examples:

E_1 = (mammalian A) & (bred-by-man A)
E_2 = (domestic B) & (viviparous B)

where (mammalian x) is a predicate which is true if x is a mammalian. (bred-by-man x) is true if x is bred by a man, (domestic x) is true if x is a domestic animal, and (viviparous x) is true if x is viviparous.

Suppose also that you know the three following rules:
R_1: $\forall$ x [(mammalian x) & (bred-by-man x) $\Longrightarrow$ (domestic x)]
R_2: $\forall$ x [(domestic x) & (viviparous x) $\Longrightarrow$ (mammalian x)]
R_3: $\forall$ x [(domestic x) $\Longrightarrow$ (inoffensive x)]
where (inoffensive x) is true if x is in general, inoffensive.

How would you generalize the two examples? How can we control the use of the axioms, without applying all the possible axioms on the examples?

A generalization of E_1 and E_2 is:
(mammalian x) & (domestic x)
and we shall see in paragraph 3.3. how we can learn it.

From now on, we suppose that we have n examples to generalize, and that we are at step 3 of the algorithm of Structural Matching, that is to say that we have chosen a constant in each example and replaced it by a VGE, named x, and we are interested in trying to make the discriminating occurences of x disappear.

3. The elimination of Discriminating Occurences

3.1. Definitions

Consider now an example E_1 and an occurence of x in that example. That occurence is completely determined by the atom (P cv_1 ... x ... cv_n) of E_1 in which it occurs and by its position in that atom. It is important to also give the position of the occurence in the atom, because a same atom can have several occurences of x. In our notation, cv_i represents either a constant or a VGE for, at a previous step, it is possible that we chose a constant which occured in that atom and renamed it with a VGE.

Definition 1
Let E_1 and E_2 be two examples, and occ_1 (respectively occ_2) an occurence of x in E_1 (respectively E_2). The occurences occ_1 and occ_2 are respectively given by the atoms $atom_1$=(P_1 cv_{11} ... x ... $cv_{n1,1}$) and $atom_2$=(P_2 cv_{12} ... x ... $cv_{n2,2}$) and by their positions p_1 and p_2 in these atoms.
We say that the occurence occ_2 **matches the occurence** occ_1 if $atom_1$ and $atom_2$ are formed with the same predicate, i.e. $P_2 = P_1$ and if the occurences have the same position, i.e. $p_1 = p_2$.

This definition is symmetrical: if occ_2 matches occ_1 then occ_1 matches occ_2 too. We say that occ_1 and occ_2 match structurally.

Consider the two following examples:
E_1 = *(book A) & (pen x) & (on A x) & (near B x)*
E_2 = *(book x) & (on x TABLE) & (near C x)*
Let occ_1 *be the occurence of x in the atom (near B x) of* E_1, *and* occ_2 *the occurence of x in the atom (near C x) of* E_2. occ_1 *and* occ_2 *match structurally.*
The occurence of x in E_1 *defined by (pen x) and the occurence of x in* E_2 *defined by (book x) do not match structurally, because the atoms, in which they occur, do not have the same predicate.*
The occurence of x in E_1 *defined by the atom (on A x) and the occurence of x in* E_2 *defined by the atom (on x TABLE), do not match structurally, because x do not occur at the same position in these atoms.*

Definition 2
Suppose that we have a set E of n examples $E_1, \ldots, E_n$.
Consider an example E_i of E and an occurence occ_i of x in E_i.
We say that occ_i is a **discriminating occurence** of x in E_i if there exists an example E_j, $j \neq i$, of E such that no occurences of x in E_j match occ_i.
E_j is called a **critical example** of occ_i.

Suppose that we have three examples:
E_1 = *(book A) & (pen x) & (on x A)*
E_2 = *(book x) & (on x TABLE)*
E_3 = *(book x) & (paper B) & (on B x)*
The occurence occ_1 *of x in* E_1, *defined by (pen x), is a discriminating occurence, because there are no occurences of x either in* E_2 *or in* E_3 *whose predicate is "pen", and hence which match this occurence. Its critical examples are* E_2 *and* E_3.
The occurence occ_2 *of x in* E_1 *defined by (on x A) is a discriminating occurence, because the only occurence of x in* E_3, *whose predicate is "on", is defined by (on B x) and x is not at the same place as in* E_1. E_3 *is the only critical example of* occ_2

This example shows us that we can see two kinds of critical examples for a given discriminating occurence: predicate critical examples and position critical examples.

Definition 3
Suppose that we have a set E of n examples $E_1, \ldots, E_n$.
Consider an example E_i of E and a discriminating occurence occ_i of x in E_i. The occurence occ_i is determined by the atom (Q cv_1 ... x ... cv_n) in which it appears, and its position in that atom.
Consider E_j a critical example of occ_i.
We say that E_j is a **predicate critical example** of occ_i, if no occurences of x in E_j occur in an atom whose predicate is Q.

In our previous example, E_2 *and* E_3 *are predicate critical examples of* occ_1.

Definition 4
With the same hypothesis as in definition 3, we say that E_j is a **position critical example** of occ_i, if there exist in E_j occurences of x whose atoms are formed with the predicate Q but which never have the same position as occ_i.

In our previous example, E_3 *is a position critical example of* occ_2.

3.2. Using the axioms

We suppose that we have a discriminating occurence occ_i of the VGE x in the example E_i and we want to make it disappear. The occurence occ_i is defined by its atom (P $cv_{i,1}$ ... x ... $cv_{i,n}$) and its position p_i in this atom.

In a critical example, no occurences of x match occ_i and we try to transform an occurence of x in order to put it into Structural Matching with occ_i.

We may think that it will be simpler to treat a position critical example of occ_i than a predicate one, since it exists in a position critical example, occurences of x, whose atoms are formed with the same predicate P as occ_i and therefore which seem rather similar to occ_i. So, in a first step, we deal with the position critical examples.

– First step:

Consider E_x, a position critical example of occ_i.

This example, E_x, contains atoms (P $cv_{j,1}$ ... x ... $cv_{j,n}$), but x is never in the position p_i in these atoms.

We try to apply axioms like commutativity, which invert the position of the variables.
The most general form of such axioms is:
$\forall x_1, \ldots, x_n \quad [(\mathrm{P}\, x_1 \ldots x_n) \Longrightarrow (\mathrm{P}\, x_{\sigma(1)} \ldots x_{\sigma(n)})]$
We must take care when we apply such axioms, especially if there are already VGEs in the atom (P cv_1 ... cv_n). If at a previous step, we introduced a VGE named y, to go on we have put all the occurences of y into Structural Matching. Therefore, for each occurence of y in an atom at a given position, there exists in each other example an occurence which matches it. If we change at another step its position by applying such a rule, we must also change the position of all the matching occurences.

If we cannot apply such axioms, we treat E_x as a predicate critical example of occ_i, since it means that the presence of occurences of x in E_x, whose predicate is the same as for occ_i is not helpful.

For instance, if we have the two following examples:
E_1 = (near x B) & (left x B)
E_2 = (near C x) & (left C x) & (right D x)
All the occurences of x in E_1 and in E_2 are discriminating.
But if we know the axioms:
R_1 : $\forall x \forall y$ [(near x y) $\Longleftrightarrow$ (near y x)]
we can improve the matching of E_1 and E_2.

The example E_2 is a position critical example for the occurence of x, defined in E_1 by the atom (near x B).
But, in E_2, we can apply R_1 to transform the atom (near C x) into (near x C). We have now:

E_1 = (near x B) & (left x B)
E_2 = (near x C) & (left C x) & (right D x)

The two occurences of x in (near x B) and in (near x C) match structurally. Let us notice that the occurence of x, defined in E_2 by the atom (near C x) was a discriminating occurence. It was transformed into a matching occurence at the same time.

The example E_2 is a position critical example of the occurence of x, defined in E_1 by the atom (left x B). But the predicate "left" is not commutative. We cannot treat it at this step.

Likewise, E_1 is a position critical example of the occurence of x, defined in E_2 by the atom (left Cx) and we cannot treat it it at this step.

– Second step

We call RCE, the set of remaining critical examples of occ_i. This set is made of the predicate critical examples of occ_i, and of the remaining position ones, which have not been treated at the first step.

We try to see if there exist on the one hand, rules which can be applied to E_i with the atom (P cv_1 ... x ... cv_n), in which the occurence occ_i appear, to generate an atom (Q cv_1^i ... x ... cv_p^i) and, on the other hand, rules which can be applied to the examples E_j of RCE to generate atoms (Q cv_1^j ... x ... cv_p^j), where the occurence of x is at the same position as in (Q cv_1^i ... x ... cv_p^i).

For the same reason as in the first step, we must take care to the VGEs already present in the examples. All their occurences are in Structural Matching, and we must not introduce new discriminating occurences of them.

The new atom (Q cv_1^i ... x ... cv_p^i), generated in E_i must not be already present in E_i, since it would bring redundant information.

It is not obvious to search for an atom (Q cv_1^i ... x ... cv_p^i), which satisfies the previous conditions. Sometimes, a lot of atoms could be deduced from (P cv_1 ... x ... cv_n) and it may take too much time to verify, for each atom, if we can introduce in the other examples occurences which match the new occurence, generated by this atom. Presently, works are done to be able to eliminate quickly some atoms which could be deduced from (P cv_1 ... x ... cv_n) but which could never be deduced in other examples.

A particular case of this method is to try to deduce from each critical example, an occurence of x, which match directly the occurence occ_i, defined by the atom (P cv_1 ... x ... cv_n). This is quite simple, because we know in this case that we want to deduce from each critical example, an atom (P ... x ...).

Suppose that we have the two following examples:
E_1 = (rectangle x) & (near A x)
E_2 = (rhombus x) & (on x B)
All the occurences of x in each example are discriminating occurences.

For the occurence of x defined in E_1 by the atom (near A x), we can apply the rules:
$\forall u\ \forall v$ *[(on u v) $\Longrightarrow$ (near u v)]*

$\forall u \ \forall v$ [(near u v) $\Leftrightarrow$ (near v u)]
to transform the atom (on x B) into (near B x) in E_2. This new occurence of x match the occurence of x, defined in E_1 by the atom (near A x).

For the occurence of x defined in E_1 by the atom (rectangle x), we can apply the rules:
$\forall u$ [(rectangle u) $\Rightarrow$ (parallelogram u)]
$\forall u$ [(rhombus u) $\Rightarrow$ (parallelogram u)]
to transform the atom (rectangle x) into (parallelogram x) in E_1 and the atom (rhombus x) into (parallelogram x) in E_2.

Therefore, we transform E_1 and E_2 into
E_1' = (parallelogram x) & (near A x)
E_2' = (parallelogram x) & (near B x)

There are no longer any discriminating occurences.

Remarks:

– This last example shows us, that often when we treat a discriminating occurence in an example, we often treat at the same time, discriminating occurences of other examples.
For instance, when we treat the occurence of x defined by the atom (near A x) in E_1, we dealt at the same time with the discriminating occurence of x, defined in E_2 by the atom (on x B). We treat also the occurence of x defined in E_2 by the atom (rhombus x), at the same time as we treat the occurence defined by the atom (rectangle x).
We modify some discriminating occurences, according to our needs for the discriminating occurences, previously chosen. To choose a discriminating occurence rather than another may lead to a different result.

Moreover, when we apply an axiom, which is not an equivalence, we loose information. If we had the occurence occ_i defined by the atom (P cv_1 ... x ... cv_n) and if we apply the axiom (P u_1 ... x ... u_n) $\Rightarrow$ (Q v_1 ... v_p), we loose the fact that there was an occurence of x in an atom whose predicate is P. We may need that occurence of x for another discriminating occurence in another example.

To avoid this, when we apply a rule which is not an equivalence, we do not replace (P cv_1 ... x ... cv_n) by (Q ... x ...) but replace it by (P cv_1 ... x ... cv_n) & (Q ... x ...), and we mark P to remember that this occurence of x in the atom (P cv_1 ... x ... cv_n) has already been dealt with. When all the discriminating occurences of x in all the examples have been treated, we can then drop the marked predicates.

– If occ_i is a discriminating occurence and if E_x is a critical example of occ_i, we search for occurences which can be transformed to match occ_i among all the occurences of x, and not only among the discriminating ones. If we find in E_x an occurence defined by (Q ... x ...), which is not discriminating and which can be transformed in (P ... x ...) to match occ_i, we use idempotency and replace (Q ... x ...) by (Q ... x ...) & (P ... x ...).

3.3. **Example**

Let us consider now the examples in paragraph 2.4. . We have given a generalization of them, but let us see how we can get it.

The examples are:
E_1 = *(mammalian A) & (bred-by-man A)*
E_2 = *(domestic B) & (viviparous B)*

and we know the three following rules:
R_1: $\forall x$ *[(mammalian x) & (bred-by-man x)* $\Rightarrow$ *(domestic x)]*
R_2: $\forall x$ *[(domestic x) & (viviparous x)* $\Rightarrow$ *(mammalian x)]*
R_3: $\forall x$ *[(domestic x)* $\Rightarrow$ *(inoffensive x)]*

First of all, we have to choose a constant in each example and replace them by a VGE. There is only a constant in the examples, so we choose them and replace them by the VGE x.

$E_1 = \underline{\textit{(mammalian x)}}\ \&\ \underline{\textit{(bred-by-man x)}}\ [\textit{(= x A)}]$
$E_2 = \underline{\textit{(domestic x)}}\ \&\ \underline{\textit{(viviparous x)}}\ [\textit{(= x B)}]$

We underline the discriminating occurences and we mark by a star the discriminating occurences, which have been already treated.

All the occurences of x are discriminating ones. We consider the first occurence of E_1, *defined by the atom (mammalian x). We see that we can deduce this atom from* E_2, *using the rule* R_2. *We get:*

$E_1 = \textit{(mammalian x)}\ \&\ \underline{\textit{(bred-by-man x)}}\ [\textit{(= x A)}]$
$E_2 = \underline{\textit{(domestic x)}}\ \&\ \underline{\textit{(viviparous x)}}\ \&\ \textit{(mammalian x)}\ [\textit{(= x B)}]$

There are still three discriminating occurences defined by the atoms (bred-by-man x), (domestic x) and (viviparous x).

We consider the discriminating occurence of E_1 *defined by the atom (bred-by-man x).*
– No rule can be applied to E_2 *to make appear (bred-by-man x).*
– We can apply the rule R_1 *on* E_1 *using the atom (bred-by-man x). It generates the atom (domestic x) and there is an occurence of x in* E_2 *which match this occurence. Therefore, we apply* R_1 *and we mark with a star the atom (bred-by-man x) to remember it has already been treated.*

$E_1 = \textit{(mammalian x)}\ \&\ {}^{*}\underline{\textit{(bred-by-man x)}}\ \&\ \textit{(domestic x)}\ [\textit{(= x A)}]$
E_2: $\textit{(domestic x)}\ \&\ \underline{\textit{(viviparous x)}}\ \&\ \textit{(mammalian x)}\ [\textit{(= x B)}]$

It remains only a discriminating occurence, which has not yet been treated, the occurence defined in E_2, *by the atom (viviparous x).*
– No rules can be applied in E_1 *to make appear the atom (viviparous x).*
– The only rule, which can be applied in E_2 *with the atom (viviparous x) is the rule* R_1. *But, it would introduce the atom (mammalian x), which is already present in* E_2, *it would not bring new information.*
No rules can be applied, we mark the atom (viviparous x) to remember that the occurence of x in this atom has already been dealt with. We get:

$E_1 = \textit{(mammalian x)}\ \&\ {}^{*}\underline{\textit{(bred-by-man x)}}\ \&\ \textit{(domestic x)}\ [\textit{(= x A)}]$

*E_2 = (domestic x) & *$\underline{\text{(viviparous x)}}$ & (mammalian x) [(= x B)]*

All the discriminating occurences have been dealt with. We drop the remaining discriminating occurences, which can never be put into structural matching. We get:

E_1 = (mammalian x) & (domestic x) [(= x A)]
E_2 = (domestic x) & (mammalian x) [(= x B)]

The two examples match structurally. A generalization of them is made of the common formula and the common links between the variables. There are no common links. A generalization G is therefore:

G = (domestic x) & (mammalian x)

3.4. Classes of predicates

For each discriminating occurence occ_i, in all the critical examples of occ_i, we search for an occurence which can be transformed to match occ_i. In order to refine this search, we define classes of predicates.

Definition 5
Let R be a rule.
We write $Pred_R$, the set of predicates used in R, both in the conditions and in the action of R.

For instance, consider the rule R_1:
$\forall x$ [(grass x) & (eat y x) $\Longrightarrow$ (grass-eater y)]
then $Pred_{R_1}$ = {grass, eat, grass-eater}

Definition 6
Let R be an axiom and C a set of predicates.
We say that R is **linked to the class** C if $Pred_R \cap C \neq \phi$, i.e. if one or more of the predicates of R also belongs to C.

For instance, if C = {square, rectangle, ellipsoid}, the rule
$\forall x$ [(circle x) $\Longrightarrow$ (ellipsoid x)] is linked to C, because the predicate "ellipsoid" of this rule belongs to C.

We construct the classes as follows:
– We choose a rule R of the knowledge base and we construct the first class made up of the predicates of R, and, if R has no preconditions, of the boolean value "true".
To sum up:
If R has preconditions, $C_1 = Pred_R$
otherwise $C_1 = Pred_R \cup \{true\}$.

– If we suppose that p classes $C_1, \ldots, C_p$ have been constructed, we consider a rule RNT of the knowledge base which has not yet been treated. Two cases are possible:

– First case: RNT is linked to a class C_i.

In that case, let $C_{i_1}, \ldots, C_{i_j}$ be the classes linked to RNT. We replace these classes by the class C defined by:
if RNT has a precondition, $C = \cup_k C_{i_k} \cup Pred_{RNT}$

otherwise $C = \cup_k C_{i_k} \cup Pred_{RNT} \cup \{\text{true}\}$

– Second case: RNT is not linked to any class.

We create a new class made up of the predicates of RNT and of the boolean value "true" if RNT has no preconditions.

For instance, if we have the following axioms:
$R_1 : \forall x \ [(grass\ x)\ \&\ (eat\ y\ x) \Longrightarrow (grass\text{-}eater\ y)]$
$R_2 : \forall x \ \ (colored\ x)$
$R_3 : \forall x \ [(square\ x) \Longrightarrow (rectangle\ x)]$
$R_4 : \forall x \ [(square\ x) \Longrightarrow (rhombus\ x)]$.
we define three classes of predicates:
$C_1 = \{grass,\ eat,\ grass\text{-}eater\}$
$C_2 = \{true,\ colored\}$
$C_3 = \{square,\ rhombus,\ rectangle\}$

The classes of predicates are interesting for two reasons:
– If P_i is a predicate used in the representation of E_i and belongs to the class C and, if in another example E_j, $j \neq i$, there are no predicates belonging to C, we cannot put the atoms whose predicates are P_i into Structural Matching with atoms of E_j. We know, before using the algorithm of Structural Matching, that we cannot put the examples into Structural Matching.
– If occ_i is a discriminating occurence of E_i, whose predicate P belongs to the class C, we must search for, in the critical examples of this discriminating occurence, occurences whose atoms also belong to C. It reduces the search.

4. Conclusion

In this paper, we have shown how axioms can be used to put examples into structural matching. Several problems have not been yet treated.
We have defined classes of predicates to decrease the search among occurences and among axioms, when we have a discriminating occurence. We have not seen how we can represent axioms to improve their use.
If occ_i is a discriminating occurence and if E_x is a critical example of occ_i, we can perhaps introduce different occurences which match occ_i. How is it to be chosen one? It may happen that the first occurence found will not lead to the best generalization.
This paper gives some indications on how improving Structural Matching, but still much work is to be done.

REFERENCES

(Bollinger 86)
Thèse de troisième cycle (not yet published)

(Biermann 84)
Biermann A.W., Guiho G., Kodratoff Y. eds
Macmillan Publishing Company.1984, pp. 463-482.

(Dejong and Mooney 86)
Dejong G., Mooney R., "Explanation Based Learning : an alternative view",
Machine learning , 2, 1986.

(Ganascia 1985)
Ganascia J.G., "Comment oublier a l'aide de contres exemples",
in Comptes rendus du congres Reconnaissances des Formes et Intelligence Artificielle, AFCET, Grenoble 1985.

(Kodratoff 83)
Kodratoff Y.: "Generalizing and particularizing as the techniques of learning"
Computers and Artificial Intelligence 2, 1983, 417-441.

(Kodratoff 84)
Kodratoff Y., Ganascia J.-G., Clavieras B., Bollinger T., Tecuci G.:
"Careful generalization for concept learning"
Proc. ECAI-84, Pisa 1984, pp. 483-492.

(Kodratoff 85)
Kodratoff Y.: "A theory and a methodology for Symbolic Learning"
COGNITIVA 85, June 4-7, 1985, pp. 639-651.

(Michalski 84)
Michalski R.S.: "A theory and Methodology of Inductive Learning"
Machine Learning, an Artificial Intelligence Approach.
Michalski R.S., Carbonell J.G., Mitchell T.M. eds
Springer Verlag 1984, pp. 83-129.

(Mitchell 83)
Mitchell T.M.: "Learning and Problem Solving"
Proc. IJCAI-83, Karlsruhe 1983, pp. 1139-1151.

(Mitchell 83)
Mitchell T.M., Utgoff P.E., Banerji R.: "Learning by experimentation, acquiring and refining problem-solving heuristics"
Machine Learning, an Artificial Intelligence Approach,
Michalski R.S., Carbonell J.G., Mitchell T.M. eds, Tioga Publishing Company 1983, pp. 163-190.

(Mitchell et Al. 86)
Mitchell T, Keller R.M., Kedar-Cabelli S.T.,"Explanation Based Learning, a unifying view",
Machine Learning 1, Kluwer Academic Publishers, 1986.

(Vere 80)
Vere S.A.: "Multilevel Counterfactuals for Generalizations of Relational Concepts and Productions",
Artificial Intelligence 14, pp. 139-164, 180.

OBJECT ORIENTED GENERALIZATION : A TOOL FOR IMPROVING KNOWLEDGE BASED SYSTEMS

M. MANAGO (mvm@lri.UUCP)

U.A. 410 du CNRS, Laboratoire de Recherche en Informatique
Bâtiment 490, Université de Paris-Sud
91405 ORSAY Cedex

ABSTRACT.
This paper is a presentation of the "object-oriented" generalization algorithm MAGGY. It is inspired by (and reacts to certain deficiencies of) the "predicate-oriented" generalization algorithm AGAPE which was developped in our research group. MAGGY outputs the minimal conjunctive generalizations of a set of positive examples. By looking at each negative example individually, it then specialize or regeneralize when needed.

I. Theoretical background.

In this paper there will be no attempt to give a formal logical definition of generalization as it is done, for instance, in [Kodratoff & Ganascia 84, Bollinger 86]. Nevertheless, we will give an intuitive definition of what generalizing means for us

I.A. Intuitive definition of generalization.

The intuitive set definition [Vere 80, Sammut 81] of generalization is as follow :

"A set X is more general than a set Y if Y is included in X."

It is difficult to prove automatically that a set is included in another one when considering variables (links between variables), properties of the connectives and rules (meta-rules) of generalization (which might take into account domain specific knowledge).

For example, one can see that G : (x ISA POLYGON) & (x ISA CIRCLE) is a generalization of E1 : (x ISA SQUARE) & (y ISA CIRCLE) & (y ISA RED-OBJECT)

Using this definition, it is possible to use the dropping rule (A is more general than A & B) on some attributes of an object. *For instance, the atomic formula (y ISA RED-OBJECT) in E1 has been dropped.* Nevertheless, it is not possible to drop all the attributes of an object (this amounts to dropping the object itself).

According to our definition, the formula G : (x ISA SQUARE) is not a valid generalization of E1 : (x ISA SQUARE) & (y ISA SQUARE). This is due to the fact that we try to compare a set of pairs (E1) with a set of singletons (G).

When the semantics of E1 is "there are two squares x and y, x May Be the Same y [Hayes-Roth 78]", E1 is a generalization of G. Indeed, we can use the idempotency of the logical & and transform G into (x ISA SQUARE) & (x ISA SQUARE) which is clearly a specialization of (x ISA SQUARE) & (y ISA SQUARE) [(x MBS y)]. Hence, the dropping rule used on objects is not a valid rule of generalization (this is actually due to the fact that the MBS link is an implicit disjunction).

In our formalism, we consider that the semantics of E1 and G are "there are at least two distinct squares" and "there is at least a square". It is intuitively clear that G is a generalization of E1. Therefore, we allow the dropping of objects in MAGGY. Note that a formal definition of generalization which allows the dropping of objects involves constructing projections into a space of lower dimension (with fewer objects).

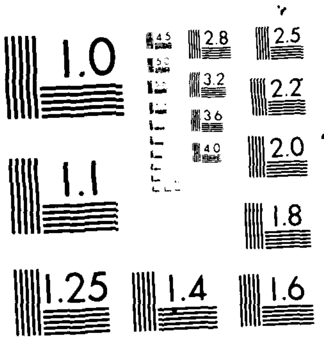
I.B. Concept learning.

We define a **primary concept** as a set of instances which give the value TRUE to an <u>atomic formula</u>.

We define a **concept** as a set of instances which give the value TRUE to some <u>conjunction of atomic formulae</u>
Consider the conjunctive formula: (x ISA BRICK) & (y ISA BRICK) & (z ISA

AD-A173 886 PROCEEDINGS OF THE INTERNATIONAL MEETING ON ADVANCES IN LEARNING HELD IN..(U) CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE ORSAY (FRANCE) L.. D J GILMORE ET AL. 2/3

UNCLASSIFIED 01 AUG 86 RR-290 F/G 5/10 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

BRICK) & (x IS-SUPPORTED-BY y) & (x IS-SUPPORTED-BY z) & ¬(y TOUCH z). The set of triplets (A,B,C) which make this formula TRUE when substituted for x, y and z, is a concept which we call an arch [Winston 75].

We define these formulae with unquantified variables to be **recognition functions**. They split the universe of instances into 3 classes : the instances which give the value TRUE to the formula, the ones which give the value FALSE and the ones which give the value UNKNOWN. It is important to remember that in this paper, a recognition function is always a conjunctive formula.

From a list of positive and negative examples of concepts, MAGGY finds concepts which are more general than the positive examples and which reject each negative example. This is done in three steps:

1) find a list of minimal (as specific as can be) conjunctive generalizations of the positive examples. MAGGY finds the generalizations in a specific order according to some heuristics (see next section). We call this module the **generalizer**.

2) Test the list of generalizations against the negative examples. The first one which is consistent (it rejects every negative example) is passed to the regeneralizer. If none is consistent, one is passed to the specializer. We call this module the **critic**.

3a) The generalization is specialized until a set of consistent recognition functions is obtained. We call this module the **specializer**.

3b) The expression is generalized again until a consistent recognition function which is maximal (as general as can be) is obtained. We call this module the **regeneralizer**.

In a knowledge based system using production rules (P_1 & P_2 &...& $P_n \Rightarrow C$), for a given conclusion C, the positive examples are the preconditions of the rules concluding to C. The negative examples are the preconditions of all the other rules. After obtaining a consistent set of generalizations of the positive examples G_1 ... G_m we can generate the production rules :

$G_1 \Rightarrow C \cdots G_m \Rightarrow C$ [Michalski & Chilauski 80]

In term of rules, the generalizer and the specializer are used to generate fewer rules which reach the same conclusions. The regeneralizer is used to remove redundant preconditions in the rules.

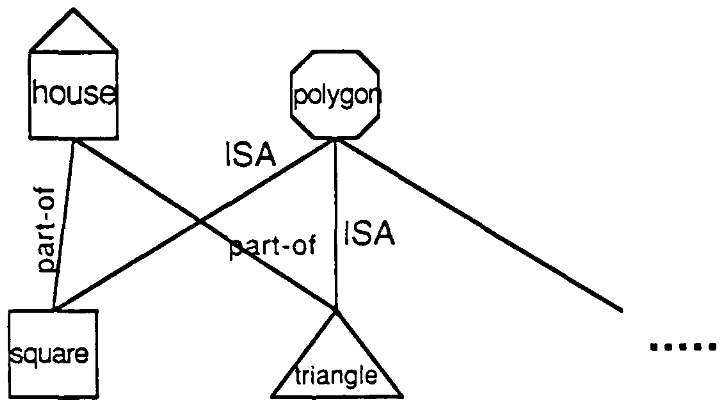
We will see in section II.B that MAGGY also learn that new primary concepts are needed to enrich the language of description.

II. MAGGY.

II.A. The generalizer.

II.A.1. Knowledge representation.

MAGGY uses ordered networks to represent relations of generality between predicates. These are frames [Minsky 75] with SS (subset) and PART-OF slots. A predicate can have more than one father, the sons are not necessarily a partition of their father (the concept they represent can overlap and they do not have to exhaust all the possibilities of their father) and there are different kinds of links of generality. This improves very much the classical taxonomic representation.



To represent the examples, MAGGY uses an object-like language with variables. *This allows to represent a RED-SQUARE in term of the primary concepts square and red-object: (x ISA SQUARE) & (x IS RED).* This representation improves the efficiency of the system. It is possible (as it is done in PLAGE [Gascuel 86]) to generate every possible combinations of primary concepts and have RED-SQUARE appearing in the frames as subsets of SQUARE and RED. Nevertheless, when there are many attributes, the combinatorics rapidly become out of hand. By using a language of first order logic, we are not faced to this problem. We only use the primary concepts which were given by the user. To simplify the notation, we will regroup all the attributes of an object. *The previous example would be rewritten as [x : (ISA SQUARE) (IS RED)].*

II.A.2. Matching objects.

MAGGY is a program which identifies sub-descriptions of examples which can be generalized. The object matcher decides which objects are going to be matched according to some heuristics and the structural matcher match these. We will not describe in here structural matching (see [Vrain & all 86]). The heuristics of generalization used by the object matcher are (in this order):

- Minimize the use of the dropping rule on objects
- Minimize the use of the dropping rule on attributes
- Minimize a conceptual distance between attributes (minimizes how high we climb the generalization frames)
- Favor some user given attributes

Consider the following examples :

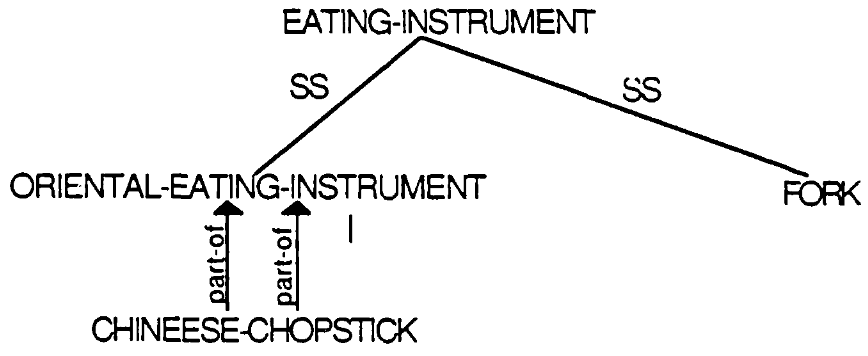
E1 : [x (ISA SQUARE) (IS RED)] & [y (ISA TRIANGLE) (IS GREEN) (IS-ON x)]
E2: [x (ISA SQUARE) (IS GREEN)] & [y (ISA TRIANGLE) (IS RED) (IS-ON x)]

The generalizer finds the two minimal conjunctive generalizations:

G1: [obj1 (ISA SQUARE) (IS COLORED)] & [obj2 (ISA TRIANGLE) (IS GREEN) (IS-ON obj1)]
G2: [obj1 (ISA POLYGON) (IS RED)] & [obj2 (ISA POLYGON) (IS GREEN)]

which mean respectively "there is a colored square and a colored triangle and the triangle is on the square" and "there is a red polygon and a green polygon". G1 is found first as it minimize the number of attributes dropped.

Consider now the following example



E1: [x : (ISA CHINEESE-CHOPSTICK) (IS RED)] & [y: (ISA CHINEESE-CHOPSTICK) (IS RED) (IS-IN GOLD)]
E2: [x: (ISA FORK) (IS RED) (IS-IN GOLD)]

The generalizer finds the generalizations:

G1: [obj1: (ISA EATING-INSTRUMENT) (IS RED)]
G2: [obj1: (PART-OF obj2) (IS RED) (IS-IN GOLD)] & (obj2 ISA EATING-INSTRUMENT)

This because G1 minimizes the dropping of objects.

The generalizer uses an "intelligent" depth-first search algorithm. This could be improved by maintaining an agenda for the backtracking. As we eventually try every possible match for an object, combinatorics could become a problem. In our application (plant pathology) it is not a problem since there are, on the average, few objects which can be matched different ways.

Furthermore, the combinatorics is reduced by forbiding dropping certain attributes. For instance, attributes describing relations between objects ((ON x), (NEAR x), (LOVE x), etc...) and certain special attributes which give a type to the object. For example, in plant pathology, it does not make sense to match a green leaf and a yellow spot since LEAF and SPOT belong to different frames and cannot be dropped (we use a user filled CANNOT-DROP slot in the frames in order to know the attributes that can be dropped).

The generalizer preserves and enhance certain features of AGAPE [Vrain & all 1986] which are structural matching and adjunction of links between predicates.

From the descriptions "there is a blue triangle with a blue spot" and "there is a yellow square with a yellow spot", the generalizer finds the generalization "there is a colored polygon with a spot of the same color".

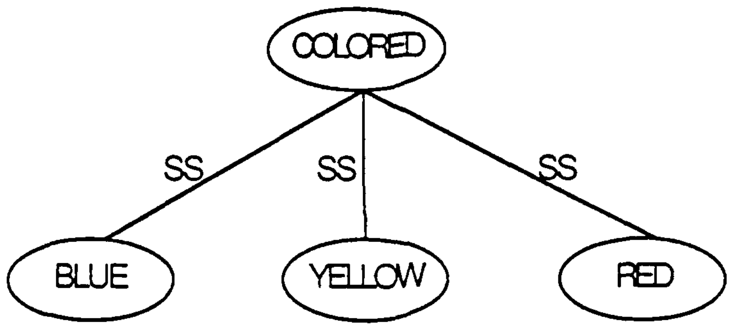
Of course, depending on the application there can be other links between predicates such as different, odd-number and so on. These links are not always relevant. Nevertheless, they might turn out to be essential to obtain a generalization which rejects the negative examples and they should not be dropped as a side effect of the algorithm (as it is the case in most bottom-up generalization algorithms).

II.B. The critic.

The modules described in the rest of this article have not yet been implemented. This will be done shortly.

The critic sequentially compare the minimal generalizations produced by the generalizer with each negative example taken individually. The first generalization which does not cover any negative example is passed to the regeneralizer. If none is found, the critic try to explain why the generalization covers some negative examples. This is done by observing which attributes of the positive examples have been over-generalized or should have their value filled in. The best result is obtained when we find a negative example close the positive examples (a near miss [Winston 75]).

Consider the following example :

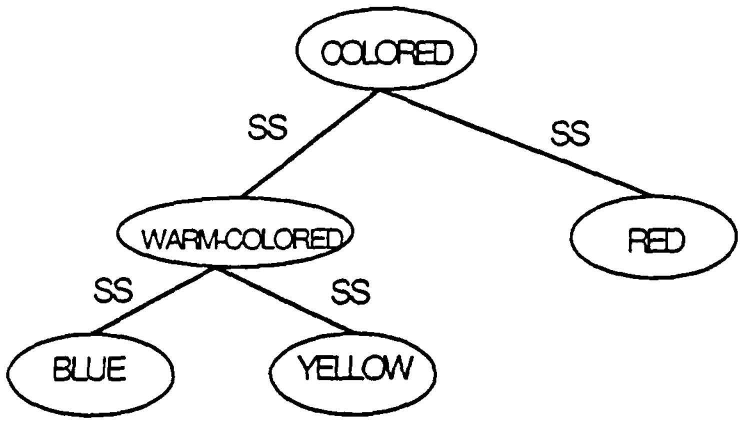


Let E1and E2 be examples of a concept, and CE a near miss.

E1: [x: (IS RED) (ISA SQUARE)]
E2: [x: (IS YELLOW) (ISA SQUARE)]

CE: [obj1: (IS BLUE) (ISA SQUARE)]

MAGGY outputs the generalization G: [obj1: (IS COLORED) (ISA SQUARE)] which covers CE. MAGGY offers to the user to create a new primary intermediary concept which is an ancestor of YELLOW and RED but not of BLUE. The color frame is then interactively changed into:



We then obtain the generalization G': [obj1: (IS WARM-COLORED) (ISA SQUARE)] which rejects CE.

The critic also attempt to fill up missing NOT in the descriptions of the positive examples. Indeed, a human being usually forgets to mention the attributes which must be false in the descriptions of the examples. This is illustrated in the arch example:

Let G be a generalization of the arch concept.

G: (obj1 ISA POLYHEDRAL) & [obj2: (ISA BRICK) (SUPPORT obj1)] & [obj3: (ISA BRICK) (SUPPORT obj1)]

and let CE be a near miss of this concept

CE: (obj1 BRICK) [obj2: (ISA BRICK) (SUPPORT obj1)] & [obj3: (ISA BRICK) (SUPPORT obj1) (TOUCH obj2)]

CE is covered by G. This is due to the fact that we are using the dropping rule on the attribute TOUCH(y). As TOUCH(y) takes the value UNKNOWN in G, we offer the user to add to the descriptions of the positive examples ¬(obj3 TOUCH obj2) which will appear in G (note that we first have to check if twe did not drop the attribute TOUCH(y) in one of the examples when we generalized).

The critic also uses default values to fill in forgotten information. If one of the default attributes of an object discriminate the negative example which is covered, we automatically fill in the value for this attribute in the generalization. The default attributes are found in the frames.

II.C. The regeneralizer.

The regeneralizer is similar to Buchanan's RULEMOD module in META-DENDRAL [Buchanan 78], AGAPE's over-generalization [Clavieras 84] or Michalski's extention against the negative examples [Michalski 83]. It removes attributes (or objects) which are not discriminant and generalize as much as possible the discriminant ones as shown in the following example:

E1: [x: (ISA SQUARE) (IS RED)]
E2: [x: (ISA SQUARE) (IS BLUE)]
CE1: [x: (ISA STAR) (IS GREEN)]
CE2: [x: (ISA CIRCLE) (IS PURPLE)]

After generalization we obtain G: [obj1 (ISA SQUARE) (IS COLORED)]. Assuming that the CANNOT-DROP slot of COLORED is set to NIL, we obtain after regeneralization G': (obj1 ISA POLYGON).

The information about which attributes are discriminant is provided to the regeneralizer by the critic who explains why G does not cover the negative examples. It is obtained by putting the positive and the negative examples in structural matching and analysing the differences between the expressions.

II.D. The specializer.

The specializer is a top down algorithm. By specializing the expression provided by the critic, it rejects the negative examples until it obtains a set of consistent generalizations of the positive examples. The goal is to minimize the cardinal of this set (it obviously lies in between 1 and the number of positive examples). This is illustrated by the following example:

E1: [x: (ISA SQUARE) (IS RED)]
E2: [x: (ISA SQUARE) (IS GREEN)]
E3: [x: (ISA CIRCLE) (IS GREEN)]

CE: [x: (ISA CIRCLE) (IS RED)]

The expression given to the specializer by the critic is G: [obj1: (ISA CONVEX) (IS COLORED)] which covers CE. The explanation is that the ISA and IS attributes of the positive examples have been over-generalized.

The specializer "un-generalize" one of the attributes which was over-generalized. This attribute is choosen according to the following

heuristics (in that order) :
- maximize the number of negative examples which will be rejected by specializing the attribute
- maximize the size of the clusters of positive examples
- use the heuristics of generalization mentionned in section II.A.2.

In the previous example, chosing either attributes leads to the same result. The critic accept both solutions which are:

G1: [obj1: (ISA SQUARE) (IS COLORED)]
E3: [obj1: (ISA CIRCLE) (IS GREEN)]

G1: [obj1: (ISA CONVEX) (IS GREEN)]
E2: [obj1: (ISA SQUARE) (IS RED)]

The specializer actually allows to construct consistent packages of positive examples (these are not necessarily clusters as the packages can overlap each others). When these packages are found, the critic pass the recognition functions to the regeneralizer.

III. Perspectives for the future.

The weak feature of MAGGY is that it cannot yet deal with incorrect descriptions (noise). For a detailed discussion of what noise can be, see [Kodratoff & all 86]. We will try to solve some of the problems relating to noise by injecting top down reasoning (à la ID3 [Quinlan 83]) into MAGGY. Bringing numeric learning techniques (that can cope with noise very well) into a symbolic learning program will, in our opinion, be very benefical to both methods.

ACKNOWLEDGMENTS

This work has been partially supported by Greco and PRC Intelligence Artificielle, by ESPRIT contrat INSTIL 1063 and by Apollo computer. It was done in Dr Yves Kodratoff's research group at the Université d'Orsay Paris-sud. MAGGY is written in LELISP on a VAX 750. It will soon be reimplemented in common lisp on an Apollo DN3000 work-station.

REFERENCES

Buchanan 78] Buchanan, B. G. and Feigenbaum, E. A.
DENDRAL and Meta-DENDRAL: their applications dimension
Artificial Intelligence 11, 1978.

[Buchanan & Shortlife 84] Buchanan, B. G. and Shortlife, E. H.
Rule-based expert systems Addison-Westley Publishing
Compagny Inc., Reading, Massachussetts 1984.

[Bollinger 86] Bollinger, T.
Généralisation en apprentissage à partir d'exemples
Thèse de troisième cycle, Université d'Orsay Paris-Sud 1986.

[Clavieras 84] Clavieras, B.
Modification de la représentation des connaissances en
apprentissage inductif. Thèse de troisième cycle, Université
d'Orsay Paris-Sud 1984.

[Ganascia 85] Ganascia, J.G.
Comment oublier à l'aide de contre-exemples
Proceeding of the 5th AFCET conference, Grenoble 1985.

[Gascuel 86] Gascuel, O.
PLAGE: A way to give and use knowledge in learning.
Proceedings of the European Working Session on Learning 1986,
Université d'Orsay Paris-sud 1986.

[Hayes-Roth & McDermott 78] Hayes-Roth, F., McDermott, J.
An interference matching technique for inducing abstractions
C. ACM 21, 1978, 401-411.

[Kodratoff & Ganascia 83] Kodratoff Y., Ganascia J.-G.
Improving the generalization step in learning
Proc. International Machine Learning Workshop, Monticello 1983.

[Kodratoff & all 86] Kodratoff Y., Manago M.,Smallman S., Blythe J., Andro T.
Integration of numeric and symbolic techniques in learning
submitted for publication, AAAI workshop Banf Canada.

[Fu & Buchanan 85] Fu Li-Min, Buchanan B. G.
Inductive knowledge acquisition for rule-based expert systems
Knowledge system laboratory, Computer Science department
Stanford University.

[Michalski & Chilauski 80] Michalski, R. S. and Chilauski, R. L.
Learning by being told and learning from examples: an experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis.
Policy analysis and information systems, Vol 4 No 2 June 1980.

[Michalski 83] Michalski, R.S.
Theory and Methodology of Inductive Learning Michalski R.S., Carbonell J.G., Mitchell T.M. eds, "Machine Learning, an Artificial Intelligence Approach." Tioga Publishing Company 1983.

[Minsky 75] Minsky, M.
A framework for representing knowledge. P.H. Winston ed, "The psychology of computer vision", Mc Graw-Hill, New York 1975.

[Mitchell 78] Mitchell, T. M.
Version Spaces: An approach to concept learning.
PhD thesis, Stanford University, December 1978.

[Quinlan 83] Quinlan, J. R.
Learning efficient classification procedures and their application to chess end games. Michalski R.S., Carbonell J.G., Mitchell T.M. eds, "Machine Learning, An Artificial Intelligence Approach", Tioga Publishing Compagny 1983.

[Sammut 81] Sammut, C.
Learning concepts by performing experiments. Dept of Computer Science, University of New South Wale Australia, nov 81

[Vere 80] Vere, S. A.
Multilevel counterfactuals for generalizations of relational concepts and production rules. Artificial Intelligence J. 14,1980, 139-164.

[Vrain & all 86] Vrain C., Manago M., Ganascia J. G., Kodratoff Y.
AGAPE: An algorithm that learn from similarities. Proceedings of the European Working Session on Learning 1986, Université d'Orsay Paris-sud 1986.

[Winston 75] Winston, P.H.
Learning Structural Descriptions from Examples. Winston, P. H. ed, "The psychology of computer vision" McGraw-Hill, New York 1985.

MACHINE LEARNING AND META-LEVEL INFERENCE

Stefek J. MALKOWSKI Zaba

HEWLETT PACKARD - Filton Road Stoke Gilford BRISTOL BS126QZ

ENGLAND

-:-

Machine Learning and Meta-level Inference

Abstract

The technique of meta-level inference has been reported in the literature as having a number of benefits arising from the strong separation between object and meta levels, corresponding to "what" and "how" knowledge respectively. We investigate the impact on and scope for machine learning when such a separation is adopted. We analyse some existing machine learning work in this light, looking in particular at some recent work of Steels and van de Welde on learning in second-generation expert systems. We argue that a more satisfactory account of that work is given by a reconstruction using the techniques of meta-level inference.

0. Introduction

The technique of meta-level inference [Bundy79] separates knowledge of a domain from knowledge required to derive the solution of particular problems in that domain. By virtue of this separation, it may become possible to use a highly expressive notation to describe the domain, while preventing combinatorially explosive search by using additional knowledge to guide the search for solutions to particular classes of problems.

In this paper, we describe the technique of meta-level inference, suggest that it provides a general framework for the organisation of knowledge-based systems, and look at the effect of adopting the separation of levels on machine learning. We then look in some detail at the Second Generation Expert Systems work of Steels et al [Steels85a,85b], which is representative of a wider class of systems employing deep models as a supplement to shallow associations, and argue that a reconstruction using the techniques of meta-level inference will allow a more principled account of this system, and its learning component in particular.

1. What is meta-level inference?

Meta-level inference is a framework for designing and implementing knowledge based systems. Its fundamental principles are the separation of object level from meta level, and the use of inference at the meta level to cause problem-solving inference at the object level.

In terms of symbol manipulation, the object level defines a language and rules for manipulating sentences of that language. The meta level defines another language, the constants of which include sentences of the object level language and the rules of object level symbol manipulations. The meta level has its own rules for manipulating sentences of the meta level.

For this symbol manipulation to be useful in designing and implementing knowledge based systems, we must ascribe meaning to the symbols and manipulation rules at each language level. Typically, the object level will describe the objects and relations in the domain: informally, we may call this "what" knowledge. The meta level contains additional knowledge *about* the object level which catalogues and organises it in order to effectively solve particular problems in the domain: we may call this "how" knowledge.

The outline above has talked only of symbols and ascribing meaning to them, avoiding any commitment to particular representations - frames, production rules, or logic-based notations. The most theoretically satisfactory account of both symbol manipulation and ascribing meaning to such symbols - semantics - is, we feel, provided by the work on formal logic. Paraphrasing [McCarthy69], the apparatus of formal logic provides us with a powerful tool for establishing the *epistemological* adequacy of a given notation: that is, for deciding whether a body of knowledge represented in a given notation is capable in principle of deriving a given conclusion. The *heuristic* adequacy of a notation - that is, its ability to effectively and efficiently derive that conclusion in practice - is not addressed by formal logical analysis. The swing away from logic-based notations in AI work has, we feel, been caused by the heuristic rather than the epistemological inadequacy of these notations. We are investigating how the use of formally well-founded logical notations at both object and meta levels can recover heuristic adequacy for particular subclasses of the possible object level *deductions*.

The remainder of this section expands this introduction to the concepts, and describes some of the advantages claimed for this organisation of knowledge.

1.1 Description of meta-level inference

The basic idea of meta-level inference is the clear separation of factual information about a problem domain from control information which can guide reasoning with that information. The object level is factual in the sense that it is a theory: it is a symbolic representation in some logic that captures an idealisation of the world. An object level theory can state both facts and rules about the domain of interest. The rule *All men are mortal* is an example of an object level rule. Facts and rules must be represented in some logic which defines a syntax of well-formed formulae.

A logic also defines ways in which well-formed formulae can be combined together to produce new ones: that is, it defines how new factual information can be derived from what is known. These ways of combining formulae are called the inference rules of the logic, and are not to be confused with the factual rules written in the logic. It is an inference rule that allows us to combine *Socrates is a man* and *All men are mortal* to deduce *Socrates is mortal*.

The object level theory can be used to answer problems about a domain in the following way. The theory contains a number of axioms - facts and rules that economically capture the behaviour of the domain in a general way. To these we add hypotheses which describe the particular problem and a description of the goal statement that we require. By a process of theorem proving we then try to derive the goal statement from the axioms plus hypotheses. The point here is that the theory contains all the information necessary in principle to state the problem and derive an answer.

Unfortunately, the blind application of the rules of inference that the object level logic provides us with is insufficient in practice to find the answer, since in anything beyond a trivial theory the blind application of the inference rules is combinatorially explosive and produces overwhelmingly more useless conclusions than useful ones. Efforts to invent clever control procedures (called uniform proof procedures) that would work on *all* theories in a logic have failed. They failed because when we reason with theories we use a great deal of knowledge to guide our reasoning. This knowledge is particular to the problem domain of the theory in question and is used to select the facts and rules that will take part in inference steps.

Adding the control

What is a required is a way of formalising the knowledge used to guide the search for a proof. Meta-level inference techniques allow this to be done by creating a theory of the conditions under which various object level facts and rules should be used. Because these statements are *about* an object level theory they belong to a *meta*theory of the domain. We can then reason *about* the object level theory by reasoning *in* the metatheory. What this means is that inference at the meta-level manipulates the object theory as a data object according to the object level rules of inference as a side effect of its own reasoning.

We can conceptualise this "proof by side-effect" as the metatheory being a higher level abstraction of the domain. Metatheories express such ideas as *to solve this sort of problem first break it into the following steps*, and *always try to solve this sort of sub-problem first*. From a system point of view however the meta-level lies beneath the object level and can be thought of as an interpreter for it. This combination of conceptual abstraction and control by interpretation produces a system of considerable flexibility and power in problem solving.

1.2 Benefits of meta-level inference

Epistemological hygiene

The advantages of separating logic from control for understandability, maintainability, knowledge acquisition, and re-usability have been argued strongly by many workers in the field, including [Kowalski79] and [Clancey83].

Soundness of object-level inference

Since problems are stated and solved solely within the object level, the control heuristics of the metatheory cannot introduce errors of commission. The metatheory constrains the combination of object level formulae by object level rules of inference to produce new object level formulae. If the object level rules of inference are sound, the object level proof induced by such a metatheory will be sound. The metatheory will, however, usually be incomplete with respect to all the possible consequences of the object level formulae and rules of inference. This makes more precise the earlier idea of "particular subclasses of the possible object level deductions" for which we can hope to "recover heuristic adequacy".

Reduction of search space

The metatheory induces only a subset of the object level inferences sanctioned by the object level rules of inference: this is the primary justification for introducing the metatheory! Of course, the writer of the metatheory must be careful not to replace an intractable combinatorial explosion at the object level with another one at the meta level. In currently implemented examples of meta-level inference such an explosion has been avoided. These examples have reduced the search space by introducing a vocabulary in which abstractions of many object level inferences relevant to the problems addressed by the metatheory can be concisely expressed.

2. The generality of the meta-level architecture

Meta-level inference as described above can be seen either as a *technique* - yet another specialised tool in the practising artisan knowledge engineer's toolkit, and probably an artefact of a (misguided) decision to use formal logic directly for knowledge representation - or as a *methodology* and *architecture*, implying that it can be used to gain insight into knowledge based systems in general. In the rest of this paper we examine some existing machine learning techniques and systems in terms of the architecture, and argue that useful insights do indeed result.

3. Machine Learning within the meta-level architecture

In this section we look firstly at the scope for machine learning within meta-level inference: that is, *where* learning techniques can fit. We then look at the need for machine learning; that is, *why* machine learning techniques may be appropriate, and what problems of the architecture they can help with. We cite examples of existing machine learning work as concrete instances of the type of learning we describe.

3.1 The scope for learning

The division between object and meta levels which the architecture enforces suggests an obvious way of classifying opportunities for learning by the level at which they learn. We take such a division as a starting point for describing learning mechanisms which might be appropriate within this architecture, but as we will see, it is valuable to make more detailed distinctions than simply object versus meta level.

Learning at the object level only

It is clearly possible to use learning by example methods such as ID3 [Quinlan79] and focussing [Young77] as aids to knowledge acquisition. Such techniques produce classification rules given a set of examples. To be used within the architecture, these classification rules would have to be interpreted by an appropriate metatheory. This need not restrict their applicability, as it is possible to transform the generated decision tree into a variety of forms by straightforward logical manipulations.

Learning at the meta-level

Various forms of learning can be conceived of to acquire or extend a metatheory. At one extreme, by treating the logic of the meta-level as a domain in its own right, we can imagine using the same example-driven techniques as could be used at the object level. Such example-driven techniques could be used in at least two ways.

Firstly, we might try to learn the definition of one of these meta-level predicates simply by being given examples and non-examples of object level entities and an indication of whether the predicate holds for them.

Secondly, we might wish to learn the conditions under which a particular object level proof step should be taken. We would need examples classified as appropriate and inappropriate instances of the proof step. Each such example would additionally need, as attributes in ID3 terminology, values for meta-level predicates chosen in advance to describe the object level sentences, the object level inference rule, and the object-level goal towards which progress is to be made. In simpler cases the goal might always be the same (e.g. "solve the equation in a small number of steps") and the concept of "useful conditions under which to take such a proof step" would not need to refer to the object-level goal state in its definition.

The correct choice of meta-level attributes is critical to the success of this approach, as is always the case with example-driven methods. The "modifier" component of Mitchell's earlier LEX work, described in the first part of [Mitchell84], can be seen as an instance of this example-driven approach.

Contrasted with the use of example-driven techniques is the use of analytic learning techniques. When used to learn an improved metatheory, these techniques analyse more closely the steps of the object level proof which the existing meta-level has induced in the object level. Such an analysis always requires a representation of the object-level goal state towards which the proof steps are intended to lead. This can be seen most clearly in Mitchell's later LEX-2 work (foreshadowed in [Mitchell82] and partly described in the second part of [Mitchell84]). Silver's LP (Learning Press) system [Silver85] similarly analyses "worked examples" of equation solving: these are sequences of object-level proof steps supplied by a benign teacher, which LP generalises to acquire meta-level knowledge about how to apply similar sequences of object-level rewrite rules in order to solve similar equations. LP is able to make very useful generalisations from a single example, by having a great deal of knowledge about the structure of equation solving. It is not clear to what degree LP has an explicit representation of what it means to be a solved equation, although we note that IMPRESS [Bundy85] certainly has such a representation.

Two features which distinguish these two forms of learning at the meta-level are:

1 The source of the *classification* of the object level entities. In the first case distinguished above, this was done by a "black box" critic which fed all the data to a concept-learning algorithm which has no knowledge of the use to which the concept it is learning will be put. In the latter case, an *analytic or deductive* learning technique is used to perform this classification with reference to the meta-level specification of the object-level goal state: the concept being acquired will be used in an implementation of that specification.

2 The *metalanguage vocabulary* used to describe the object level entities. In the first case distinguished above, this is again supplied from without. At the other extreme, the learning component may indeed be creating new terms in this metalanguage.

A more comprehensive treatment of these distinctions appears in [Boswell85].

3.2 The need for learning

Having outlined the possible applications of machine learning techniques within the architecture, we consider what benefits arise from their use. We see two potential shortcomings of the architecture presented so far which machine learning can address. One concerns the assumption that the subset of object-level queries which are to be answered efficiently is known in advance, so that a specialised metatheory can be hand-crafted for that subclass. The second concerns the assumption that adequate problem-solving performance can always be achieved by a suitably guided series of object level inferences.

Dynamic improvement of a metatheory

In practice, we may be unable or unwilling to predict characteristics of the queries which will be posed to a given object theory, and so we might provide an initially inefficient metatheory which in use would be improved by a learning component to optimise performance on the queries which do occur most frequently. Techniques which could come under this broad heading range from simple result cacheing, through learning the meta-level properties of the object-level entities most heavily used in practice, all the way to learning new meta-level concepts and using them to improve the metatheory.

What if smart control isn't enough?

It may be that the object level theory is so large or badly structured that even cleverly guided object level inference is too inefficient in practice to perform the required inferences at an acceptable speed. An extreme and artificial example would be an object level theory consisting only of the Peano axioms for the natural numbers, which we were attempting to use to find the prime factorisation of (large) numbers; another one is Kowalski's presentation of "Slowsort" [Kowalski79]. Faced with this situation, we can adopt one of two strategies, depending on whether the class of queries to be more efficiently solved can be described in advance.

We may try to learn a new object theory - meta theory pair with the same input-output behaviour as the old object theory - meta theory pair. The danger here is if we treat the old theories as a black box, we lose the soundness property we have claimed for the meta-level architecture, as we will have no justification for the inferences performed in the new theories. The second-generation expert systems work of Steels and van de Welde [Steels85a, Steels85b] appears to be an instance of this technique: this work is examined in more detail in the following section.

Or, we may *compile* a specialised object - meta theory pair, suitable only for solving the restricted class of problems which the original metatheory solved, but incorporating the choices the old metatheory had to make dynamically for each query into a form where the choices are made in advance. The object theory resulting from such a compilation may need only a very simple metatheory indeed to interpret it - such as an unaugmented Prolog interpreter. The KARDIO work of Lavrac, Bratko et al [Lavrac85] can be seen as an instance of this technique.

4. Relation to Second-Generation Expert Systems

It is interesting to look at the learning component of the work reported in [Steels85a] and [Steels85b] in the light of the meta-level architecture. The "second generation expert systems" described are diagnostic expert systems, consisting of not only surface heuristic rules relating symptoms to causes and (trivially) remedies, but also a deep reasoning component which comes into play when the remedy recommended by the surface rules does not clear the symptom. The surface rules are not present at all when the system first starts: they are learned as the results of successful diagnosis in the deep model.

The deep model

The deep model is intended to completely capture the design knowledge concerning the system to be diagnosed. In the references, the example systems are subsystems of a car. The design knowledge represented describes the function of the overall system as the functioning of its mponent parts. The design knowledge is held in the form of a network such as Figure 1.

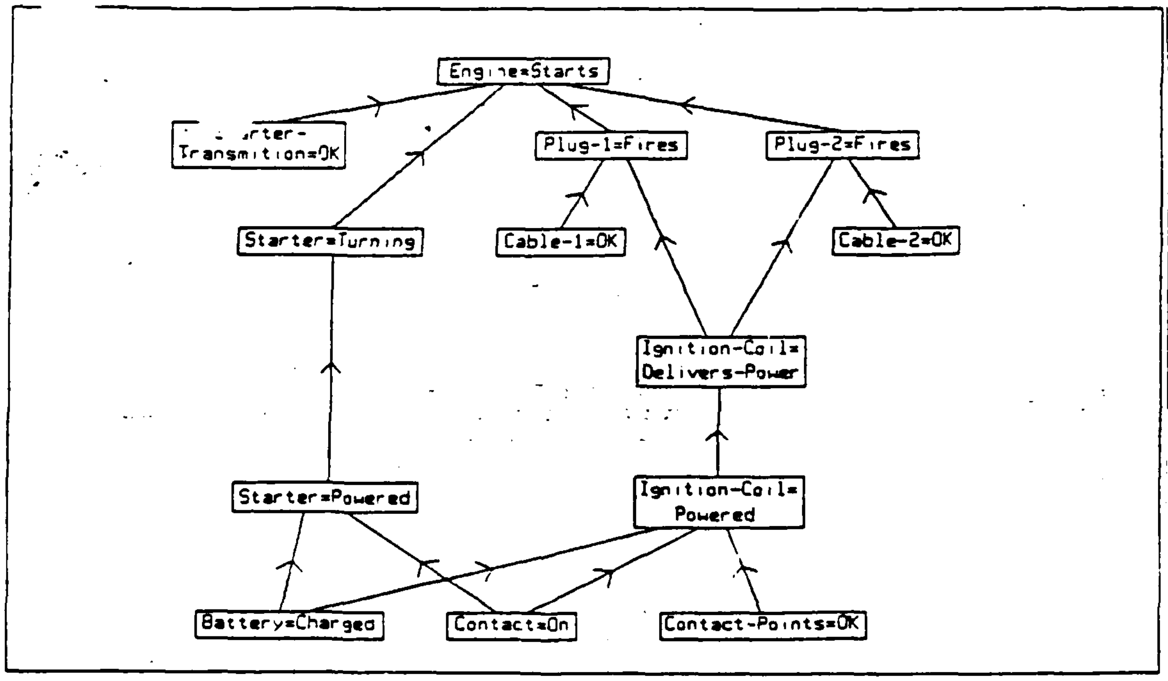


Figure 1 (reproduced from [Steels85b]

We can easily translate this network into a propositional logic form. Each node is a proposition interpreted as the correct functioning of a component. The directed arcs represent individually necessary and collectively sufficient conditions for the correct functioning of the component towards which a set of arcs is directed. For example, the topmost node and its incoming arcs can be represented as

```
Engine_Starts <->
   Starter_Transmission_OK & Starter_Turning & Plug1_Fires & Plug2_Fires
     & Engine_Function_OK
```

The Plug1_Fires proposition is further refined as

```
Plug1_Fires <->
   Cable1_OK & Ignition_Coil_Delivers_Power & Plug1_Function_OK
```

Note that in performing this reconstruction, an extra proposition representing the correct internal functioning of each component has been conjoined with the propositions representing the incoming arcs. This makes explicit in the object level representation the possibility of component failure which appears to be buried in the causal network interpreter in the original work.

Problem solving in the deep model is performed by noting "anomalies" - differences between the desired and actual values of the propositions represented by the nodes - and inferring the values of propositions to which this "anomalous" proposition is equivalent. The many possible solutions are pruned by requiring the values given to the propositions to be consistent with other observed propositions.

In terms of the meta-level architecture, the causal model is a complete object level theory of the functioning of the car's ignition system. The deep problem-solving strategy is a metatheory which controls the application such object level rules of inference as:

$$\begin{array}{l} X \leftrightarrow A \,\&\, B \,\&\, C \\ \sim X \\ \hline \quad \sim A \vee \sim B \vee \sim C \end{array} \qquad \text{and} \qquad \begin{array}{l} X \leftrightarrow A \,\&\, B \,\&\, C \\ X \\ \hline \quad A \,\&\, B \,\&\, C \end{array}$$

These are applied in a relatively exhaustive way in order to find a consistent assignment of values to propositions. In domain terms, these rules respectively justify the reasoning principles stated as "if the effect of a property is anomalous, it may itself be anomalous too" and "if the effect of a property is normal, then there is reason to believe that it itself is normal", by reading "a property" as a chosen one of A, B, or C, and "its effect" as X.

The metatheory is augmented with domain-specific "inspection rules" which control where in the causal network this reasoning starts. It seems that the actual deep reasoning metatheory used may also distinguish "external controls" and "observable properties" among the propositions which represent the various components, and implicitly embody such metatheoretic principles as "test values of observable properties and external controls before attempting to test values of internal components".

The surface rules

Despite incorporating this metaknowledge, this metatheory is expensive to run and causes the system to consider many possible component failures. The system therefore further attempts to learn "shallow heuristic associations between observable symptoms and their probable ultimate causes". These heuristic rules summarise a series of inferences in the causal model, and are known as "solution rules". It is vital to note that the source of candidate surface rules is not only the deep causal model, but also an indication from the real world of whether the remedial action suggested by a diagnosis - whether from the existing shallow rule-set or from the deep model - succeeded or failed. The "solution rules" are composed of:

* the "primary symptom" - a proposition whose falsehood triggers the use of the solution rule
* "secondary symptoms" - propositions which must be true for the rule to be used
* "corrected properties" - propositions whose falsehood is asserted to imply the falsehood of the primary symptom, and which when made true imply the truth of the primary symptom.

An example pair of solution rules is:

Primary-symptoms: NOT Car_Starts
Secondary-symptoms: Gas_Present
Corrected-Properties: Parking_Mode

Primary-symptoms: NOT Car_Starts
Secondary-symptoms: NOT Gas_Present
Corrected-Properties: Gas_Present

We are told to read these rules respectively as follows:

IF the car doesn't start AND gas is present, THEN put the car in parking mode.
IF the car doesn't start AND there is no gas present, THEN make gas be present.

It appears to be possible to represent the object level content of these surface rules in a propositional logic form related to that given for the causal network. One possible translation schema is

Secondary-Symptoms -> (Corrected-Properties <-> ~ Primary-Symptoms)

which can be read as "if the secondary symptoms are present, the presence of the primary symptoms is explained by the absence of the corrected properties, and making the corrected properties true will make the primary symptoms false", that is, cure the primary symptoms. Applying this translation schema gives

Gas_Present -> (Parking_Mode <-> Car_Starts)
~ Gas_Present -> (Gas_Present <-> Car_Stars)

the second of which simplifies to Car_Starts -> Gas_Present, which can be read as Gas_Present being a necessary condition for Car_Starts.

In the absence of any explicit semantics for the "solution rule" syntax, it is possible to try various other translations of the solution rules. Different translation schemas would sanction various aspects of the behaviour of the rules and the rule integration procedure which combines solution rules deemed to be "incompatible". Translations which deal only with the object level content of the rules suggest that the shallow rules are an unjustifiable transformation of the deep model. Such translations, however, obscure the main point of these solution rules, which is to function as problem-solving heuristics. We argue below that the shallow rules and their interpreter are in fact encoding, in an obscure way, valuable problem solving knowledge.

What is really being learned?

It appears that this program works as suggested at the end of section 3.2. The metatheory and object theory are judged to be too inefficient, and so a different theory is constructed which exhibits some of the same input-output behaviour. However, it is not merely the efficiency of the metatheory which is at issue: it may also suggest *undesirable actions* on the basis of the object level inferences.

For instance, given only the general inference rule that a device will misfunction if it is faulty itself or if its enabling conditions are not all true, it might naively suggest that the ignition coil function be checked - which would involve dismantling part of the engine and testing the coil with specialised equipment - before checking whether the battery is charged.

Such behaviour can in favourable circumstances be avoided by the surface rules: if a flat battery is correctly diagnosed as being the cause of an ignition failure on one occasion, they will suggest this diagnosis alone in future. In the event of a subsequent ignition failure being cured by replacing the ignition coil, the surface rules might still recommend that the battery charged be checked before replacing an ignition coil, although the actual order in which the surface rules would recommend the actions would appear to depend critically on the details of the "rule integration" procedure.

What is missing, then, from the naive metatheory interpreting only the causal model, is any notion of the *relative cost of tests* and the *likelihood of individual faults*. The overall learning component is attempting to learn the *consequences* of this knowledge, in the form of the surface-level rules. Consistently with our definition of an object level theory, the deep model is capable only of predicting all *possible* faults, not the *likely* ones. This is the knowledge which is being learned learn from the environment. The learning component of this system, however, produces as output the solution rules shown above, in which the relative likelihoods and orderings are hidden in the *orderings of these rules* and of their conditions.

It is helpful here to bring in Clancey's classification of the types of knowledge a diagnostic system such as Mycin contains - implicitly or explicitly [Clancey83]. In those terms, we have:

Support knowledge	the deep causal model
Strategic knowledge	the diagnostic metatheory (interpreter of the causal model). This metatheory may contain some references to knowledge to guide the diagnosis, but such knowledge is not sufficiently available
Structural knowledge	the meta-level classification of object level terms. This is the missing link, which is being learnt from the environment.

To illustrate the above claim, we believe that the diagnostic metatheory needs to be able to express such principles as:

* suggest COMMON causes before RARE ones
* perform CHEAP tests before EXPENSIVE ones

However, these principles cannot guide the inference in the absence of meta-level clauses such as

```
commonly_false( park_mode ).
commonly_false( gas_present ).
cost_of_testing( park_mode, 0 ).
cost_of_testing( ignition_coil_OK, 1000 ).
```

This clearly identifies why the simple exhaustive metatheory fails to produce expert problem solving behaviour, and suggests we should concentrate on acquiring and explicitly representing the information the environment makes available. Indeed, having made explicit the knowledge which is missing, we note that some of it could be explicitly acquired from a practising expert, or indeed deduced from an even more detailed deep model.

Another disadvantage of compiling the problem solving ability of the deep model and the diagnostic metatheory is that the deep model becomes unavailable to support any other tasks. For instance, the surface rules cannot even in principle answer many of the questions identified in [Kidd85] as being frequently asked of human experts, such as "why did that remedy work?", or "will remedy X work?". By encoding he "structural" knowledge in an obscure way, it also becomes impossible for the system to explain its diagnostic strategy adequately: it cannot say "you should look for fuel in the tank before taking the fuel pump to pieces because it's easier."

5. Conclusion

We have described an architecture based on *meta-level inference* and argued that it provides a sound basis for work in machine learning. Its close relation to formal logic, the best tool available for describing the manipulation of symbols and ascribing meaning to those manipulations, makes it more clear exactly *what* is being learnt. The *explicit representation* of control knowledge enforced by the architecture is particularly valuable when control is being learned, and is greatly preferable to encoding such knowledge implicitly in the order and structure of arbitrary symbol-structures whose semantics are unclear. We have illustrated this point by looking in some detail at the work of Steels et al. We claim that a reconstruction of that system within the meta-level architecture is feasible and desirable. Such a reconstruction would exhibit equivalent behaviour and give a more satisfactory account of it.

Acknowledgements

Many of my colleagues at HPLabs Bristol have contributed to the ideas presented in this paper. I am particularly indebted to Bill Sharpe for introducing me to the technique of meta-level inferance, and to John Lumley, Martin Merry, and Bob Welham for commenting critically on a draft of this paper at very short notice. The formulaic nature of the traditional disclaimer that all remaining errors and shortcomings are my own responsibility in no way reduces its truth. Finally, I gratefully acknowledge the support of Hewlett-Packard in providing the time and facilities for producing this paper.

References

[Boswell85]
Boswell, R. *An Analytic Survey of Analytic Concept-Learning*, Edinburgh University Dept. of AI Working Paper no. 181.

[Bundy79]
Bundy, A., et al. "Solving Mechanics Problems using Meta-Level Inference", in *IJCAI-79*, pp.1017-1027.

[Bundy85]
Bundy, A., & Sterling, L., "Meta-Level Inference in Algebra". Available as Edinburgh University Dept. of AI Research Paper no. 273, 1985; to appear in *Proceedings of the Capri-85 conference on AI*, North-Holland.

[Clancey83]
Clancey, W.J., "The Epistemology of a Rule-Based Expert System - a Framework for Explanation", *Artificial Intelligence* **20**, North-Holland, 1983, pp. 215-251

[Kidd85]
Kidd. A.L., "What do Users Ask? - Some Thoughts on Diagnostic Advice", *Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems*, Merry M. (ed.), Cambridge University Press, 1985, pp. 9-19

[Kowalski79]
Kowalski R., *Logic for problem solving*, Elsevier, New York, 1979, pp. 125-127

[Lavrac85]
Lavrac, N., Bratko, I., Mozetic, I., et. al. "KARDIO-E - an expert system for electrocardiographic diagnosis of cardiac arrhythmias", *Expert Systems*, 2, 1 (January 1985), Learned Information, Oxforf, England

[McCarthy69]
McCarthy, J., & Hayes, P., "Some Philosophical Problems from the standpoint of Artificial Intelligence", *Machine Intelligence 4*, Metzler B. & Michie D. (eds.), Edinburgh University Press, 1969, pp. 463-502. Reprinted in *Readings in Artificial Intelligence*, Webber B.L. & Nilsson N. (eds.), Tioga Publishing Co., Palo Alto, 1981

[Mitchell82]
Mitchell, T.M., "Toward Combining Empirical and Analytic Methods for Learning Heuristics," *Human and Artificial Intelligence*, Elithorn, A. & Banerji, R. (eds.), Erlbaum, 1982.

[Mitchell84]
Mitchell, T.M., Utgoff, P.E., Banerji, R., "Learning by Experimentation: Acquiring and Refining problem-solving heuristics", *Machine Learning: an AI Approach*, Michalski, Carbonell, & Mitchell (eds.), Springer-Verlag, Berlin, 1984, pp. 163-190

[Quinlan79]
Quinlan, J.R., "Discovering rules by induction from large collections of examples", *Expert Systems in the Micro-Electronic Age*, Michie D. (ed.), Edinburgh University Press, 1979, pp. 168-201

[Silver85]
Silver, B. *Precondition Analysis: Learning Control Information.* EDAI Research Paper nr. 220, Dept. of A.I., Edinburgh University, 1985.

[Steels85a]
Steels, L., & van de Welde, W., "Learning in Second Generation Expert Systems", in *Knowledge-Based Problem Solving*, Kowalik J.S. (ed.), Prentice-Hall, New Jersey, 1985

[Steels85b]
Steels, L., "Second Generation Expert Systems", *Future Generation Computer Systems* 1 (4), 1985, pp.213-221

[Young77]
Young, R., Plotkin, G.D., & Linz, R.F., "Analysis of an extended concept-learning task", in *IJCAI-77*, p.285

Learning Fault Diagnosis Heuristics from Device Descriptions

Michael J. Pazzani
The Aerospace Corporation
P.O.Box 92957
Los Angeles, CA 90009

Introduction

This paper describes an approach to learning efficient heuristics for diagnosing faults in complex systems. This technique is applicable to the learning heuristics for the identification of failures of components of large systems whose status is monitored for unusual or atypical features, such as a power plant or a satellite. When one or more atypical features are detected, a diagnosis process seeks to find an explanation for the atypical features. This explanation typically involves isolating the cause of the atypical features to the failure of a component. Occasionly, the explanation may be that system is in a normal but unusual mode[1]. The focus of our investigation is the attitude control system of the DSCS-III satellite.[2] The system is implemented in a combination of LISP and PROLOG.

Two different approaches have been used for fault diagnosis. In one approach [3, 7, 20], the observed functionality of devices are compared to their predicted functionality which is specified by a quantitative or qualitative model of the device [4, 8, 6]. For a large system whose status is changing rapidly, comparing observed to predicted functionality can be costly. The alternative approach [23, 16, 25] encodes empirical associations between unusual behavior and faulty components as heuristic rules. This approach requires extensive debugging of the knowledge base to identify the precise conditions which indicate a particular fault is present. In previous work, [17, 18] we have described the Attitude Control Expert System (ACES) in which these two approaches are integrated. Heuristics examine the atypical features and hypothesize potential faults. Device models confirm or deny hypothesized faults. Thus, heuristics focus diagnosis by determining which device in a large system might be at fault. Device models determine if that device is indeed responsible for the atypical features.

In this paper, we address the problem of revising the fault diagnosis heuristics when they hypothesize a fault which is later denied. This occurs when all of the possible exceptions to a heuristic are not explicitly stated. When a fault is proposed, and later denied by device models, the reasons for this hypothesis failure are noted and the heuristic which suggested the fault is revised so that the hypothesis will not be proposed in future similar cases. This is a kind of failure-driven learning [21] which enables a diagnostic expert system to start with heuristics which indicate some of the signs (or symptoms) of a failure. As the expert system solves problems, the heuristics are revised to determine what part of the device model should be consulted to distinguish one fault from another fault with similar features. There are several reasons why this approach is desirable:

- Device models are a natural way of expressing the functionality of a component. However, they are not the most natural or efficient representation for diagnosis [22].
- Determining some of the signs of a fault (i.e., the initial diagnostic heuristics) is a relatively

[1]It is often the case that the monitor is designed to have a tolerable number of false alarms, rather than miss an actual failure.

[2]The attitude control system is responsible for detecting and correcting deviations from the desired orientation of the satellite.

easy task. For example, ACES starts with a heuristic which states that if a tachometer is reading 0, then it is faulty. Later this heuristic is revised to include conditions to distinguish a fault in a tachometer from a fault in the component measured by the tachometer.

The following example illustrates failure driven learning of diagnosis heuristics. I once noticed that the left taillight of my car was not working. I knew of two reasons that a taillight could be out which might be expressed as the following two heuristics in an expert system:

```
IF a taillight is not working
THEN the fuse of the taillight's circuit is blown.

IF a taillight is not working,
THEN the bulb of the taillight is burnt out.
```

I was able to rule out a blown fuse. If the fuse were blown, then all of the lights on the same fuse would be out. Consulting my owner's manual, I discovered that the right front parking light would also be out if the fuse were blown. The first fault diagnosis heuristic could be modified to prevent considering this hypothesis in the future:

```
IF a taillight is not working
AND the opposite front parking light is not working
THEN the fuse of the taillight's circuit is blown.
```

One way to view this type of learning is as an extension of dependency-directed backtracking [24]. In dependency-directed backtracking, when a hypothesis failure occurs, the search tree of the current problem is pruned by removing those states which would lead to failure for the same reason. In failure-driven learning, the reason for hypothesis failure is recorded, so that the search tree of future similar problems does not include states which would lead to failure for the same reason.

In the remainder of this paper, we first discuss some related work in machine learning on improving performance with experience. Next, we describe our approach to learning efficient diagnosis heuristics. Finally, we present an example of the approach applied to the attitude control system of DSCS-III.

Previous Work

R1-Soar

R1-Soar [19] is an attempt to duplicate the performance of R1 [10], an expert system that configures computers, by learning configuration strategies. R1-Soar utilizes a learning mechanism called *chunking* as implemented in Soar [9]. R1-Soar starts with an initial *base* representation which indicates the goal to be achieved and operators that can be used to achieve the goal state. In Soar, all basic operations are represented as subgoals. For example, subgoals will be spawned to select among applicable operators, to test if a goal has been achieved, and to find the result of applying an operator to a state. With the base representation and subgoaling strategy, R1-Soar can search for the solution to any configuration problem but this search may be expensive. In Soar, efficiency is achieved by rules which guide the search. These rules are automatically acquired by creating chunks of knowledge implicit in the base representation. Chunking is a technique for recording the solution of a subgoal so that the chunk can substitute for the subgoal processing the next time the same subgoal is encountered. For example, chunking a goal to select among operators will result in a chunk which selects the proper operator in that state. Chunking is accomplished by creating a new rule. The test of the rule is found by noting what facts were accessed to solve the subgoal. The action of the rule is computed by noting what facts were added to memory during the processing of the subgoal which are needed by the parent goal.

R1-Soar presents an interesting approach to learning which we share. Expert systems can be viewed as knowledge-intensive programs, as opposed to domain independent general purpose problem solving programs. Much work is required to build and debug the knowledge base of an expert system. In contrast, less effort is required to define the general knowledge needed by a problem solver, but the general problem solver is bound to be more inefficient since it must search for a solution. A primary difference between R1-Soar and our work is the mechanism that creates a knowledge-intensive expert system from a general problem solver. R1-Soar uses a general technique which records the solution to every subgoal. One question unanswered by R1-Soar is when learning is beneficial. Clearly, it is not valuable to remember that on March 4, 1984 at 10:35 the momentum was within normal bounds. Even though it may take 50 primitive operations to recalculate this fact, it is not worth learning since it will not be used again. Our more specific approach only learns one thing: how to avoid making the same mistake. An additional problem with Soar is that it can over-generalize. For example, R1-Soar learned that a module could not be put in any backplane, where it should have learned that the module could not be put in a particular backplane. Over-generalization is a serious problem which must be addressed before Soar can be used in a practical application.

Failure-driven Learning

Schank [21] has proposed failure-driven learning as the mechanism by which a person's memory of events and generalized events evolves with experience. A person's memory provides expectations for understanding natural language understanding and inferring other's plans and goals. When a new experience fails to conform to these expectations, it is stored in memory along with the explanation for the failure to prevent the generation to the erroneous expectation in the future. In Schank's theory, the reason for the expectation failure can be a Motivational Explanation (i.e., an actor is pursuing a different goal than inferred) or an Error Explanation (i.e., an actor was not able to accomplish his goal which was inferred correctly). The correction to memory so that the failure does not occur again is to remember the event causing the failure indexed by the explanation for the failure. In future similar situations, this event will be the source of expectations rather than the generalized event whose expectations were incorrect. In failure-driven learning as applied to fault diagnosis, the failures are of fault hypotheses as opposed to expectations. The reason for failure is identified as some aspect of the device's function which disagrees with the fault hypothesis. The correction is to modify the heuristic rule which proposed the incorrect hypothesis to check that aspect of the device before proposing the fault.

Explanation-based Learning

Failure-driven learning dictates two important facets of learning: when to learn (when a hypothesis failure occurs) and what to learn (features which distinguish a fault in one component from faults in other components). What is not specified is how to learn. For example, a learning system could learn to distinguish a faulty tachometer from failures with similar features by correlation over a number of examples (e.g. [11, 13, 25]). Device models (or a teacher) could classify a large number of examples as positive or negative examples of broken tachometers. For example, the heuristic which suggests broken tachometers could be revised to include a description of those combination of features which are present in a number of examples when a tachometer is faulty, but not present when the tachometer is working properly.

In contrast, ACES learns how to avoid a hypothesis failure after just one example. The conditions which need to be tested to avoid a hypothesis failure are exactly those features of the one example which were needed by the device models to deny the hypothesis. The device models serve a dual role here. First, they identify when to learn by denying a hypothesis. More importantly, they provide an explanation for the hypothesis failure. The device models indicate which features would have been needed to be present (or absent) to confirm the hypothesis. This deductive approach to learning is called **explanation-based learning** [5, 12, 14]. Explanation-based learning improves the performance of ACES by creating fault diagnosis heuristics which explicate information implicit in the device models.

Failure Driven Learning of Fault Diagnosis Heuristics

In this section, we describe our approach to learning fault diagnosis heuristics by finding symptoms of faults implicit in device models. First, let us clarify what we mean by a device model. Following Chandraskaran [22], we represent the following aspects of a device:

- **Structure:** Specifies the connectivity of a device.
- **Functionality:** Specifies the output of a device as a function of its inputs (and possibly state information).

It is not important to the expert system or the learning module that the functionality be expressed quantitatively or qualitatively. The important part is that given the observed inputs of a device, the device model can make a prediction about the output. The predicted value of the output be compared to the observed value or can be treated as an input to another device.

Reasons for Hypothesis Failure

We have identified three different reasons for failing to confirm a hypothesis. For each reason we have implemented a correction strategy.

- **Hypothesized Fault- Inconsistent Prediction:** The hypothesized failure is inconsistent with observed behavior of the system. The strategy for correction is to check for other features which the proposed fault might cause. The hypothesis failure in the example of the taillight discussed earlier is of this type.
- **Hypothesized Unusual Mode- Enablement Violated:** The atypical features can be explained by the system being in a normal but unusual mode. However, the enabling conditions for that mode are not met. For example, once the EGR (Exhaust Gas Recirculation) warning light on my car went on indicating that the emission system needs servicing. In an expert system this might be expressed:

```
IF the EGR light is on
THEN the emission control system needs service
```

  When I read owner's manual for my car, I found that the light goes on every 25,000 miles. Since the car had around 13,000 miles on it, the emission control system didn't need service (although the light did).

  The strategy for correcting the heuristic which proposed the faulty hypothesis is simply to consider one of the enabling conditions of the unusual state. In general, there may be several conditions which define such an unusual state. Only those conditions which would be true if the system were in an atypical mode but are not true in the current example are used to revise the fault diagnosis heuristic. The above rule would be changed by this strategy:

```
IF the EGR light is on
AND the odometer is near a multiple of 25,000
THEN the emission control system needs service
```

- **Hypothesized Fault– Unusual Input:** The device hypothesized to be faulty is in fact functioning properly. This typically occurs when the input to a device is very unusual. In this case, the output of the device may also be unusual and the device might be assumed to be faulty unless the input is considered. For example, when I was a young child living in New Jersey, my television stopped working; only static appeared on all of the stations. I tried to fix it by adjusting the fine tuning knob and finally asked my mother for help. My mother assumed that I had broken the tuner by twisting it so much. Apparently, she has a heuristic which might be expressed:

```
IF there is static on all stations
THEN the  tuner is broken
```

  Several hours later, she discovered that there was a power failure in New York and none of the television stations were broadcasting. The problem with this heuristic is that it doesn't consider that the input to the television might be at fault. Revising the above rule to account for the input relationship would result in the following:

```
IF there is static on all stations
AND the stations are broadcasting
THEN the the tuner is broken
```

Whenever a hypothesis is denied by consulting a device model, the reason for the denial must be found to avoid the failure in future similar cases. We have identified these three sources of hypothesis failure and use a different correction strategy for each failure.

Revising Fault Diagnosis Heuristics

When there is a hypothesis failure, the explanation for the failure is found and the heuristic rule which proposed the hypothesis is revised. A heuristic rule which proposes a fault can apply to one particular component (e.g., the light bulb of the left taillight) or a class of components (e.g., light bulbs). Similarly, the correction strategy can apply to a particular component or a class of components. The manner in which the knowledge base of heuristic rules is revised depends on the generality of the heuristic rule and correction. These interact in the following manner:

- **Heuristic rule not more general than the correction:** The correction is added to the heuristic rule and this new more specialized rule replaces the old rule.
- **Heuristic rule more general than the correction:** The correction is added to the heuristic rule and applied only in the specialized case. The old rule is retained for use in other cases.

Consider the case of the taillight discussed earlier. This example assumed that the explanation for ruling out the fuse was expressible as "If the fuse for a taillight is blown, then the front parking light on the opposite side will be out". Since the rule and the revision both applied to any taillight (i.e., same level of generality), the rule is replaced by the revised version. On the other hand, if the explanation were expressed as "If the fuse for a left taillight is blown, then the right front parking light will be out", then the new rule could not replace the previous rule:

```
IF a left taillight is not working
AND the right front parking light is not working
THEN the fuse of the taillight's circuit is blown.
```

In a similar manner, if the original fault diagnosis heuristic were expressed more generally, about car lights in general instead of about taillights, then it would need to be specialized about taillights but remain to diagnose problems with other lights. The knowledge base would then need to contain the following two rules:

```
IF a taillight is not working
AND the opposite front parking light is not working
THEN the fuse of the taillight's circuit is blown.

IF a light is not working
THEN the fuse of the light's circuit is blown.
```

There are two other issues to be considered in revising heuristic rules. First, since some testing is being added to hypothesis generation, it would be wasteful to repeat the same test during confirmatic To avoid this potential problem, the revision to a rule caches the results of a test. Second, the amount of search necessary to prove a conjunction of subgoals in PROLOG (the language we use to implement ou rules) is dependent on the order in which the subgoals are attempted. We use a strategy to order the tests in a revised rule similar to one proposed by Naish [15]. This strategy minimizes the size of the search space by detecting the ultimate failure of a rule as soon as possible. This assumes that decreasing the search space is the best means of increasing performance. This is true in our applicatio since testing for the presence or absence of any feature is equally expensive. Cantone [1] gives an approach for ordering tests based in part on the cost of the test.

A Definition of Failure-driven Learning of Fault Diagnosis Heuristics

More formally, a diagnosis heuristic can viewed as the implication:

$$F \text{ and } \mathbf{consistent}(H) \rightarrow H$$

where F is a set of features, H is a hypothesis, and **consistent(H)** is true if believing H does not result in a contradiction.[3] In our approach to learning and fault diagnosis, **consistent(H)** corresponds to confirming a hypothesis with device models. Confirmation can be viewed as the following implications:

$$H \rightarrow C_1$$
$$H \rightarrow C_2$$
$$\dots$$
$$H \rightarrow C_n$$

If F is true, but **consistent(H)** is false because not(C_i) is true, then the diagnosis heuristic can revised to:

$$F \text{ and } C_i \text{ and } \mathbf{consistent}(H) \rightarrow H$$

In some cases, checking the consistency of a hypothesis with the device models is more properly viewed as the following implication:

$$B \text{ and } H \rightarrow C_l$$

The situation when B is true and C_l is false and corresponds to the case that the revised heuristic is used in addition to the old heuristic. The form of the revised heuristic in this case is:

$$F \text{ and } C_l \text{ and } B \text{ and } \mathbf{consistent}(H) \rightarrow H$$

The point of failure-driven learning of diagnosis heuristics is that it is simpler to rule out a hypothesis by testing for C_l than proving **consistent(H)**.

[3]See [2], for a discussion of "consistent".

Failure-driven Learning of Diagnosis Heuristics

In this section, we describe an example of how the performance of the expert system to diagnose faults in the attitude control system is increased through failure-driven learning. To follow this example, it is necessary to know a little about attitude control.

Attitude Control

The attitude control system consists of a number of sensors which calculate the satellite's orientation on the three axes (called yaw, pitch and roll) by detecting the location of the earth and the sun and a set of reaction wheels which can change the satellite orientation if it deviates from the desired orientation. There are four reaction wheels (PY+, PY-, PR+, and PR-), arranged on the four sides of a pyramid (see Figure 1). Pitch momentum is stored as the sum of all four wheel speeds; roll momentum is stored as the difference between the PR+ and PR- speeds; and yaw momentum is stored as the difference between the PY+ and PY- speeds.

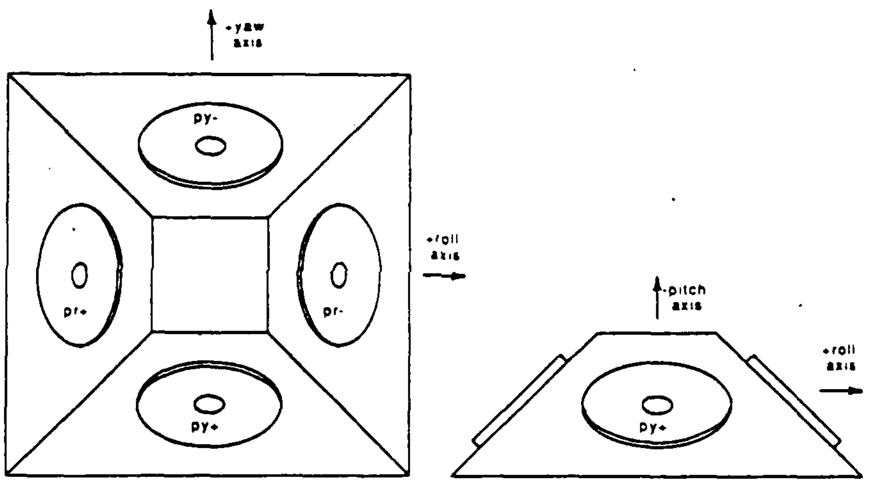


Figure 1: The reaction wheels

A diagram of the attitude control system appears in Figure 2. The signals YATT, RATT, and PATT represent the attitude on the yaw, roll and pitch axes respectively. The wheel drive signal processing component issues drive signals to the motor of the reaction wheels to change the wheel speeds. The wheel drive signals are WDPY+, WDPY-, WDPR+ and WDPR- for the PY+, PY-, PR+ and PR- wheels respectively. The wheel speeds are measured by tachometers yielding the signals WSPY+, WSPY-, WSPR+ and WSPR- for the PY+, PY-, PR+ and PR- wheels respectively. The tachometer signal processing module converts the four wheel speeds to the three values representing the equivalent wheel speeds on the yaw, roll, and pitch axes. These equivalent wheel speeds are also combined with the attitude information from the sensors to yield the estimated attitudes (YATT, RATT, and PATT).

The attitude control system contains the logic necessary to maintain the desired attitude. For example, to compensate for a disturbance on the roll axis, the difference between the speed of PR+ and PR- wheels must change.

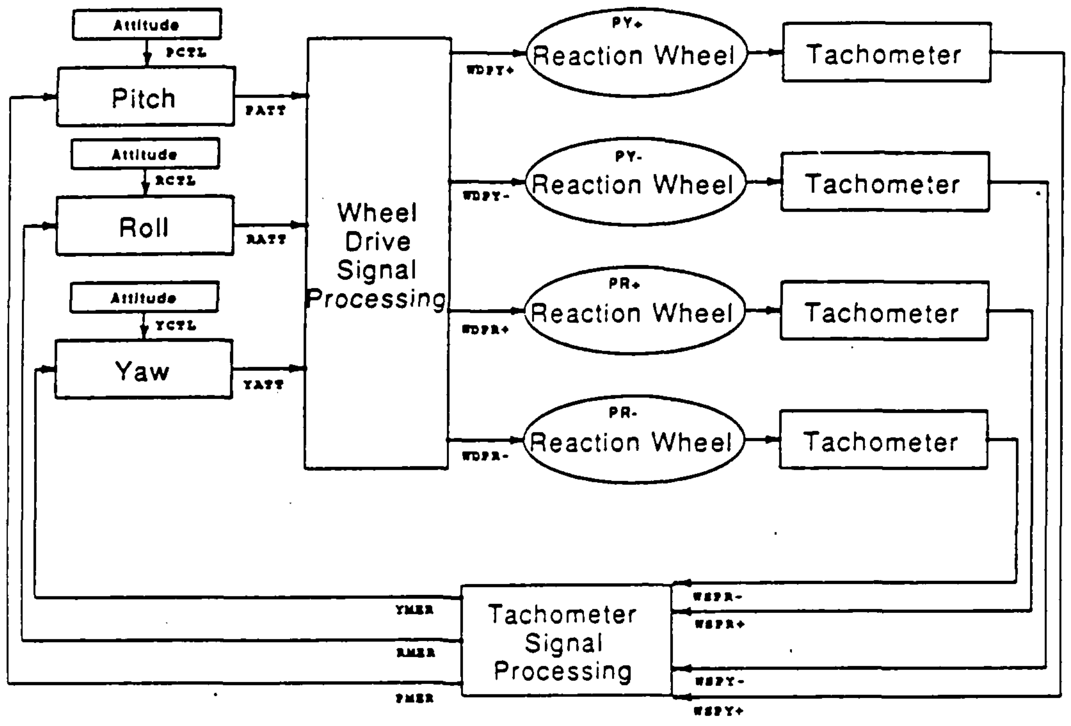


Figure 2: Block diagram of the attitude control system

ACES: The Attitude Control Expert System

One reason that our particular satellite was chosen for this research is that The Aerospace Corporation possesses a simulator for the attitude control system which generates telemetry tapes reflecting faulty behaviors to aid engineers in faults diagnosis. In addition, these tapes serve as input to our expert system. ACES consists of two major modules:

- *Monitor.* This module converts the raw telemetry data to a set of features which describe the atypical aspects of the telemetry. In ACES, the features detected include:
  - *(value-violation signal start-time end-time value)*: Between *start-time* and *end-time* the average *value* of *signal* has taken on an illegal *value*.
  - *(jump signal start-time end-time amount start-value end-value slope)*: The *signal* has changed from *start-value* to *end-value* between *start-time* and *end-time*. *Amount* is the difference between *start-value* and *end-value* and *slope* is *amount* divided by the difference between *start-time* and *end-time*.
- *Diagnostician.* This module finds an explanation for the atypical features.

In this article, we focus on the learning in the diagnostician. The diagnostician which is illustrated in Figure 3 is comprised of several cooperating modules:

- *Fault Identification.* The atypical features are used as symptoms of faults by heuristic rules to postulate a hypothesis which could account for the behavior of the satellite. Typically, the hypothesis isolates the atypical behavior to a failure of a single component.

- *Fault Confirmation.* This step compares the actual device functionality to the functionality as specified by a device model. This process either can confirm or deny that a hypothesized fault is present. If a hypothesis is denied, an attempt is made to identify another fault.
- *Fault Implication Analysis.* After a fault has been confirmed, the effect of the fault on the values of other telemetry signals is assessed. A model of the attitude control system predicts the values of telemetry signals which might be affected by the fault. The predicted telemetry values are analyzed by the monitor to see if they are atypical. Descriptions of atypical predicted values are then compared against the set of atypical features to explain any features which are a result of a confirmed fault.

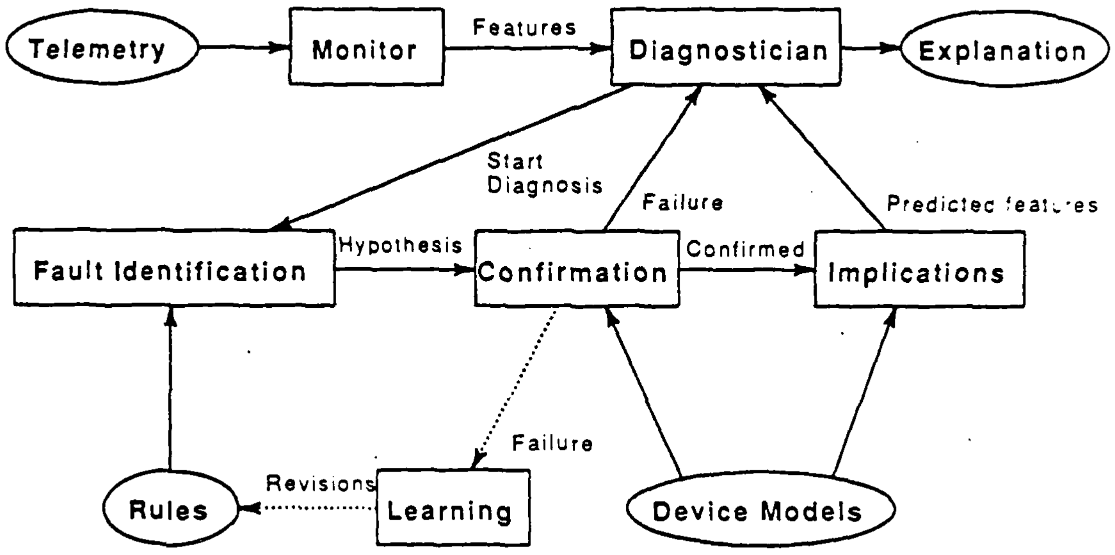


Figure 3: Block diagram of the Attitude Control Expert System

Refining Fault Diagnosis Heuristics

For this example, the initial fault diagnosis heuristics are quite simple. Figure 4 presents the definition of two fault diagnosis rules. These PROLOG rules have a LISP-like syntax since our PROLOG is implemented in LISP. The first element of a list is the predicate name. Variables are preceded by "?". The part of the rule preceded by ":-" is a fault hypothesis, and the part of the rule after ":-" are those conditions which are necessary to be proved to propose the hypothesis. These rules implement two very crude heuristics: "if the speed of a reaction wheel is 0, then the tachometer is broken" and "if the speed of a reaction wheel is 0, then the wheel drive is broken".

Since ACES is implemented in PROLOG, it tries the heuristic rules in the order that they are defined. However, for the purposes of learning, we find it more convenient to have the ordering of the rules undefined.[4] This prevents one heuristic from relying on the fact that another fault proposed by an earlier

[4]This is implemented by randomly changing the order of the rules before each run.

```
1: (problem (problem wheel-tach ?from
                    (broken-wheel-tach ?wheel ?from))) :-
;there is a tachometer stuck at 0
(feature(value-violation ?sig ?from ?until 0))
(measurement  ?sig ?wheel speed ?tach)
(isa ?wheel reaction-wheel)
;if the speed of a wheel is 0

2: (problem (problem wheel-drive ?from
                    (broken-wheel-drive ?wheel ?from ?sig))) :-
;there is a wheel drive motor not responding to the drive signal
(feature(value-violation ?sig ?from ?until 0))
(measurement  ?sig ?wheel speed ?tach)
(isa ?wheel reaction-wheel)
;if the speed of a wheel is 0
```

Figure 4: Initial Fault Diagnosis Heuristics

heuristic has been ruled out.

An example will help to illustrate some of the strategies for revising fault diagnosis heuristics. Figure 5 contains the relevant telemetry data. For this telemetry tape, the monitor notices several atypical features:

1. WSPR-, WSPR+, WSPY+ and WSPY- *have changed an unusual amount.*
2. WSPR+ and WSPR- are 0.

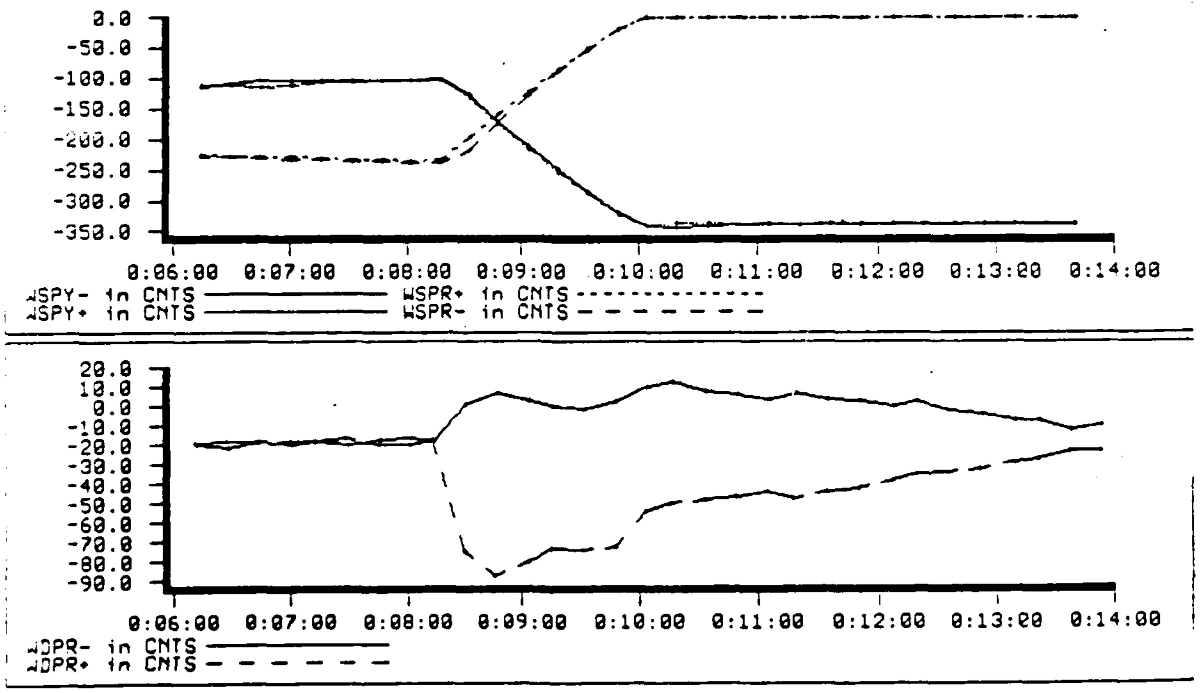


Figure 5: Telemetry data after a broken wheel drive

The first hypothesis proposed by the first rule in Figure 4 is that the tachometer of the PR- wheel is stuck at 0. The confirmation module denies this hypothesis for the following reason: if the tachometer were stuck at 0, the attitude of the satellite would change drastically.[5] Since the attitude did not change, the heuristic must be revised to avoid the generation of this hypothesis in future similar cases. The hypothesis failure is caused by not checking the implications of a faulty tachometer (Hypothesized Fault-Inconsistent Prediction). Checking any of the attitude signals would suffice to distinguish a faulty tachometer from the actual fault. In Figure 6, the revision tests YATT.

```
(problem (problem wheel-tach ?from
                  (broken-wheel-tach ?wheel ?from))) :-
(FEATURE (VALUE-VIOLATION YATT ?FROM-32 ?END-33 ?VALUE-34))
;MAKE SURE THE YAW ATTITUDE HAS BEEN DISTURBED
(feature(value-violation ?sig ?from ?until 0))
(AFTER ?FROM-32 ?FROM)
;MAKE SURE THE ATTITUDE DISTURBANCE IS AFTER THE VALUE VIOLATION
(measurement  ?sig ?wheel speed ?tach)
(isa ?wheel reaction-wheel)
(CACHE-PROVED ATTITUDE-DISTURBANCE)
```

Figure 6: Revised Faulty Tachometer Heuristic– changes in SMALL CAPITALS

After the heuristic has been revised, diagnosis continues. The next hypothesis proposed by the second rule in Figure 4 is that the wheel drive of the PR- is broken. The device model of a wheel drive includes the following information: the wheel speed is proportional to the integral of the wheel drive signal. If the wheel drive signal is positive the wheel speed should increase.

During the time that WSPR- increased from -100 to 0, WDPR- was positive (see figure 5). Therefore, the PR- wheel was not ignoring its drive signal and the hypothesis is denied. The hypothesis failure is caused by the fact that WSPR- wheel is indeed doing something very unusual by changing so rapidly and stopping. However, it is doing this because it is responding to WDPR-. The heuristic which proposed this fault is revised to consider the functionality of the device (Hypothesized Fault– Unusual Input).

In Figure 7, the revised heuristic checks that change of the wheel speed as it approaches 0 is not due to the drive signal. Since our heuristic rules and our device models are implemented in the same language, it is possible to move code from the device model to a heuristic rule by renaming variables. In other systems, this may not be possible. However, this strategy would still apply if the rule could be revised to indicate what part of the device model to check for (e.g., test that the observed wheel speed could not be produced given the wheel drive between $time_1$ and $time_2$). In ACES, it is possible to revise the rule to specify <u>how</u> the test should be performed instead of <u>what</u> test should be performed.

After the heuristic has been revised, another hypothesis is found to account for the atypical features: the faulty wheel drive heuristic proposes that the PR+ drive is ignoring its input since WSPR+ is 0, and when it increased to 0, WDPR+ was negative indicating that the speed should decrease (see Figure 5). The confirmation of this hypothesis is trivial since the heuristic already proved that the drive was not functioning according to its device description. After the fault is confirmed, the effects on the rest of the attitude control system are assessed. Since roll momentum is stored as the difference between the speed of the PR+ and PR- reaction wheels, when WSPR+ goes to 0, WSPR- should change by the same

[5]The attitude control system would believe that the wheel was not storing any momentum when in fact it is. To compensate for the erroneous report of loss of momentum, the attitude control system would adjust the momentum of the other wheels, changing the attitude of the satellite.

```
(problem (problem wheel-drive ?from
                  (broken-wheel-drive ?wheel ?from ?sig))) :-
(FEATURE(JUMP ?SIG ?FROM-37 ?UNTIL-38 ?JUMP-39 ?START-40
            ?END-41 ?SLOPE-42))
;THERE IS A CHANGE IN THE WHEEL SPEED
(feature(value-violation ?sig ?from ?until 0))
(AFTER ?FROM ?FROM-37)
;THE WHEEL SPEED REACHES 0 AFTER IT CHANGES
(measurement  ?sig ?wheel speed ?tach)
(isa ?wheel reaction-wheel)
(DRIVES ?DRIVE-43 ?WHEEL)
(MEASUREMENT ?DRIVE-SIGNAL-44 ?DRIVE-43 AMPLITUDE DIRECT)
;FIND THE WHEEL DRIVE SIGNAL OF THE ?WHEEL
(IS ?DRIVE-SIGNAL-SIGN-45
   (TELEMETRY-SIGNAL-SIGN ?DRIVE-SIGNAL-44 ?FROM-37 ?UNTIL-38))
;FIND THE SIGN OF THE THE DRIVE SIGNAL DURING THE JUMP
(IS ?SLOPE-SIGN-46 (REPORT-SIGN ?SLOPE-42))
;FIND THE SIGN OF JUMP
(NOT (AGREE ?SLOPE-SIGN-46 ?DRIVE-SIGNAL-SIGN-45))
;MAKE SURE THE DIRECTION OF THE JUMP DISAGREES WITH THE DRIVE-SIGNAL.
(CACHE-DISPROVED WHEEL-DRIVE-STATUS)
```

Figure 7: Revised Wheel Drive Heuristic– changes in SMALL CAPITALS

amount. The satellite was in a very unusual state prior to the failure: WSPR+ and WSPR- were equal. When the PR+ drive broke, WSPR- went to 0 to compensate for the change in WSPR+. In addition, since the pitch momentum is stored as the sum of all four wheels, to maintain pitch momentum WSPY+ and WSPY- decreased by the amount that WSPR+ and WSPR- increased. While WSPY+ and WSPY- decreased, the difference between them remained constant to maintain the yaw momentum. The broken PR+ wheel drive accounts for the atypical features and the diagnosis process terminates.

Results

There are two standards for evaluating the effects of learning in ACES. First, there is the performance of ACES using the rules in Figure 4. We call this version naive-ACES. Additionally, there is the performance of ACES using rules hand-coded from information provided by an expert. We call this version of the system expert-ACES. The performance of the naive-ACES after learning is compared to naive-ACES and expert-ACES in Figure 8 and Figure 9. There are four test cases which are used for comparison:

1. A tachometer stuck at 0.
2. A wheel drive ignoring its input when the opposite wheel is at the same speed. Data from this example is in Figure 5.
3. A wheel unload (i.e., the speed of the reaction wheels is changed by the firing of a thruster). This is not actually a failure, but it changes the wheels speeds and momentum so that the monitor detects atypical features.
4. A wheel drive ignoring its input in the usual case where the opposite wheel is at a different speed.

The data in Figure 8 demonstrate that the failure driven learning technique presented in this paper improves the simple fault diagnosis heuristics to the extent that the performance of ACES using the learned heuristics is comparable to the system using the rules provided by an expert. In one case, the performance of the learned rules is even better than the expert provided rules. This particular case is the previous example in which a wheel drive broke when the satellite was in an unusual state. The heuristic provided by the expert did not anticipate the rare condition that two opposing wheel speeds were equa'

CASE	fault	naive-ACES	naive-ACES after learning	expert-ACES
1	tachometer	21	1	1
2	wheel drive	4	1	2
3	wheel unload	1	1	1
4	wheel drive	2	1	1

Figure 8: Number of Fault Hypotheses

The data in Figure 9 reveal that the number of logical inferences required by the expert system decreases after learning. This demonstrates that after learning, the expert system is doing less work to identify a failure rather than moving the same amount of work from hypothesis confirmation to hypothesis generation. Comparing the number of inferences required by naive-ACES after learning to those of expert-ACES is not actually fair since it appears that the expert's rules at times test some information retested by the confirmation process. Recall that retesting is avoided by a revised rule since the revision contains information to cache the results of consulting a device model. It has been our experience that this cache reduces the number of inferences by approximately ten percent. An additional ten percent of the inferences are saved through intelligent ordering of clauses of revised rules compared to our initial simple approach of appending the revision to the end of a rule.

CASE	fault	naive-ACES	naive-ACES after learning	expert-ACES
1	tachometer	2268	211	584
2	wheel drive	1238	616	910
3	wheel unload	870	861	947
4	wheel drive	745	409	643

Figure 9: Number of Inferences to Generate and Confirm Fault

Conclusion

We have presented an approach to refining fault diagnosis heuristics by determining what aspect of a device model must be consulted to distinguish one fault from another fault with similar features. This approach relies on explaining why a heuristic does not apply in a certain case and correcting the heuristic to avoid proposing an erroneous fault hypothesis. Applying this technique to a simple version of the ACES expert system for the diagnosis of faults in the attitude control system yields performance comparable to and in some cases better than the performance of ACES with expert fault diagnosis heuristics.

Acknowledgements

Comments by Anne Brindle, Jack Hodges, Steve Margolis, Rod McGuire and Hilarie Orman helped clarify this article. This research was supported by the U.S. Air Force Space Division under contract F04701-85-C-0086 and by the Aerospace Sponsored Research Program.

References

[1] Cantone, R., Pipitone, F., Lander, W., & Marrone, M. Model-based Probabilistic Reasoning for Electronics Troubleshooting. In *Proceedings of the Eigth International Joint Conference on Artificial Inteliegence*, pages 207-211. IJCAI, Vancouver, August, 1983.

[2] Charniak, E., Riesbeck, C. and McDermott, D. *Artificial Intelligence Programming*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1980.

[3] Davis, R., Shrobe, H., *et al.* Diagnosis Based on Description of Structure and Function. In *Proceedings of the National Conference on Artificial Intelligence*. American Association for Artificial Intelligence, Pittsburgh, PA, 1982.

[4] de Kleer, J. & Brown, J. A Qualitative Physics Based on Confluences. *Artificial Intelligence* 24(1), 1984.

[5] DeJong, G. Acquiring Schemata Through Understanding and Generalizing Plans. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*. Karlsruhe, West Germany, 1983.

[6] Forbus, K. Qualitative Process Theory. *Artificial Intelligence* 24(1), 1984.

[7] Genesereth, M., Bennett, J.S., Hollander, C.R. DART: Expert Systems for Automated Computer Fault Diagnosis. In *Proceedings of the Annual Conference*. Association for Computing Machinery, Baltimore, MD., 1981.

[8] Kuipers, B. Commonsense Reasoning about Causality: Deriving Behavior from Structure. *Artificial Intelligence* 24(1), 1984.

[9] Laird, J., Rosenbloom, P., and Newell, A. Towards Chunking as a General Learning Mechanism. In *Proceedings of the National Conference on Artificial Intelligence*. American Association for Artificial Intelligence, Austin, Texas, 1984.

[10] McDermott J. R1: a Rule-Based Configurer of Computer Systems. *Artificial Intelligence* 19(3), 1982.

[11] Michie, D. Inductive Rule Generation in the Context of the Fifth Generation. In *Proceedings of the International Machine Learning Workshop*. Monticello, Illinois, 1983.

[12] Minton, S. Constraint-based Generalization: Learning Game-Playing Plans from Single Examples. In *Proceedings of the National Conference on Artificial Intelligence*. Austin, TX, 1984.

[13] Mitchell, T. Generalization as Search. *Artificial Intelligence* 18(2), 1982.

[14] Mitchell, T., Kedar-Cabelli, S. & Keller, R. *A Unifying Framework for Explanation-based Learning*. Technical Report, Rutgers University, 1985.

[15] Naish, Lee. Prolog Control Rules. In *Proceedings of the Ninth International Joint Conference on Artificial Intellegence*, pages 720-722. IJCAI, Los Angeles, CA, August, 1985.

[16] Nelson, W.R. REACTOR: An Expert System for Diagnosis and Treatment of Nuclear Reactor Accidents. In *Proceedings of the National Conference on Artificial Intelligence*. AAAI, Pittsburgh, PA, 1982.

[17] Pazzani, M. & Brindle, A. An Expert System for Satellite Control. In *Proceedings of ITC/USA/85, the International Telemetering Conference*, pages 27-41. International Foundation for Telemetering, Las Vegas, NV, October, 1985.

[18] Pazzani, M & Brindle, A. Automated Diagnosis of Attitude Control Anomalies. In *Proceedings of the Annual AAS Guidance and Control Conference*. American Astronautical Society, Keystone, CO, February, 1986.

[19] Rosenbloom, P., Laird, J., McDermott, J., Newell, A., and Orciuch, E. R1-Soar: An Experiment in Knowledge-Intensive Programming in a Problem-Solving Architecture. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 7(5), 1985.

[20] Scarl, E.A., Jamieson, J., & Delaune, C. A Fault Detection and Isolation Method Applied to Liquid Oxygen Loading for the Space Shuttle. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*. Los Angeles, CA, 1985.

[21] Schank, R. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, 1982.

[22] Sembugamoorthy, V. & Chandraskaran, B. *Functional Representation of Devices and Compilation of Diagnostic Problem Solving Systems*. Technical Report, Ohio State University, March, 1985.

[23] Shortliffe, E.H. *Computer-based Medical Consultation: MYCIN*. American Elsevier, New York, NY, 1976.

[24] Stallman, R. M. & Sussman, G. J. Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis. *Artificial Intelligence* 9(2):135-196, 1977.

[25] Vere, S. Induction of Concepts in the Predicate Calculus. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*. Tbilisi, USSR, 1975.

[26] Wagner, R.E. Expert System for Spacecraft Command and Control. In *Computers in Aerospace IV Conference*. American Institute of Aeronautics and Astronautics, Hartford, CT, 1983.

Some Predictive Difficulties in Automatic Induction

A.P. White

and

A. Reed

Centre for Computing and Computer Science,
University of Birmingham,
P.O. Box 363,
Birmingham B15 2TT

Abstract

The predictive behaviour of Quinlan's ID3 algorithm is examined when dealing with a synthetic example from number theory, in which the attributes actually convey no information about class membership. Although ID3 was able to construct rules for this subject domain, cross-validation showed them to be dramatically inadequate. Indeed, the performance was well below chance level. The reason for this is identified as use of an inappropriate branching criterion and the advantages of other branching criteria are discussed. The statistical notion of over-fitting of parameters is also introduced and the relative merits of statistical and logical approaches are mentioned.

Introduction

The basic principle behind automatic induction is well known. A problem domain consisting of a number of mutually exclusive classes is exemplified by a number of suitably chosen cases (often known collectively as the "training set"). Each such case consists of a class indicator and a vector of attribute values. A suitable induction algorithm is then set to work on these data to derive one or more rules which can then be used to classify further cases of unknown class membership (the "test set") from their attribute values.

What may be less well known in the field of Machine Learning is that there exist other techniques in the domain of Mathematical Statistics which were developed to handle more general problems of the same type. (These techniques are more general because they can handle problems in which the classes overlap in the attribute space, i.e. in which the attribute vector from a given case does not provide an unambiguous indication of the class membership of that case). These techniques yield models (often linear ones) which can then be used to classify further cases of unknown class membership. In other words, models in Mathematical Statistics are the counterpart of rules in Automatic Induction.

The Problem

Now, it is well known that statistical models are prone to a particular difficulty. It is possible to fit too many parameters so that, although the apparent fit to the test data is good, the actual predictive power of the model is actually worse than might have been achieved more parsimoniously by using a smaller number of parameters.

Note that the actual predictive power can be assessed by a process called "cross-validation". This involves employing the obtained model to classify some further cases that were not used in deriving the model. The usefulness of this approach in testing the efficacy of derived rules in the field of inductive learning is recognised and utilised (e.g. Michalski and Chilausky, 1980).

The purpose of this paper is to draw attention to the behaviour of Quinlan's ID3 algorithm (Quinlan, 1979) which, under certain circumstances, allows rules to be constructed which have no predictive power whatsoever. In fact, under these particular circumstances, cross-validation shows that the performance is actually dramatically worse than that obtainable by an intelligent guess based on the prior probabilities of class membership.

An Example

For the purpose of illustrating this particular point, we need an example about which we can be quite sure, on theoretical grounds, concerning the relationship between the attributes and the class membership. For this reason, the example actually chosen was a synthetic one, derived from number theory. It can be explained, quite simply, as follows.

The cases were actually the non-negative integers from zero to N-1, where N is some suitable power of 2. These numbers were actually regarded as falling into two classes - those exactly divisible by 3 and those not. The numbers were expressed in binary notation and each bit was considered to be an attribute.

In fact, 13 sets of consecutive integers were used. The smallest set consisted of those numbers expressible with two binary digits (i.e. the range zero to 3). The next set comprised those numbers expressible with three bits (i.e. the range zero to 7). Each successive range employed one more binary digit and hence the largest set used 14 bits and covered the range from zero to $2^{14}-1$.

Now, it is quite obvious from elementary number theory that no logical combination of bit values of a number can have any bearing whatsoever on its divisibility by 3. However, ID3 had no difficulty at all in constructing perfect classification rules for each set of integers employed.

Cross-Validation

In order to assess the true efficacy of the induced rules, a cross-validation technique was employed, as follows. Instead of using full sets of integers, as described in the previous section, all but one of the integers was used to derive the rule, which was then tested by predicting the class membership of the remaining integer. This procedure was then repeated with each integer in turn being the odd one out. This entire procedure was then repeated for each of the 13 sets of integers. (In fact there were some regularities in the results that enabled some short cuts to be taken, as well as an important symmetry which halved the work required. These are described later. However, the whole exercise still required a considerable amount of computer time). For all these tests, the window in ID3 was set wide enough to allow all the cases in a given training set to be considered at once. This was done in order to avoid possible unwanted side effects in the predictions occurring as a result of the random number generator coming into play in the selection of subsets of cases.

The class symbols 1 and 0 were used to represent, respectively, divisibility and non-divisibility by 3. Here, we shall represent the prediction outcome of actual class i and predicted class j by the ordered tuple (i,j) and the total frequency of such an outcome over a set of integers by the notation f(i,j). The corresponding probabilities are calculated from the frequency information in the usual manner and denoted by p(i,j).

Fig. 1 gives the probabilities of the various types of outcome, as a function of the number of binary digits used for the integer set. It is apparent that, from 4 bits upwards, the results for even numbers of bits were the same as those for one bit less. This is because, for each set, the first half of the prediction outcomes were the same as those for the entire set obtained from one bit less. Furthermore, with even numbers of bits, the predictions were symmetrical about the mid-point of the ordered set of integers. (These features enabled substantial savings of

No. of bits	p(0,0)	p(0,1)	p(1,0)
2	0.000	0.500	0.500
3	0.125	0.500	0.375
4	0.125	0.500	0.375
5	0.188	0.469	0.344
6	0.188	0.469	0.344
7	0.219	0.445	0.336
8	0.219	0.445	0.336
9	0.236	0.430	0.334
10	0.236	0.430	0.334
11	0.248	0.419	0.333
12	0.248	0.419	0.333
13	0.255	0.411	0.333
14	0.255	0.411	0.333

Fig. 1. Probabilities (to 3 s.f.) of prediction outcomes as a function of number of binary digits.

computing time to be made).

It will be noted that only three types of outcome appear in the results. The fourth type of outcome, i.e. (1,1), was never observed to occur in these trials. Thus the only correct predictions were for class 0. Of itself, this is not particularly surprising, as the prior probability of class membership for class 0 is twice that for class 1. However, as is apparent from the data, predictions for class 1 were made by the algorithm - but they were invariably incorrect. This is particularly noteworthy because making these less probable predictions actually reduces the chances of being correct when the attributes convey no information about class membership.

Thus the only source of correct predictions was the (0,0) events. As can be seen from the table in Fig. 1, the probability of these was remarkably low - although this probability does increase as the number of attributes is increased. The obvious question arises as to whether these probabilities are approaching some asymptotic value. Unfortunately, we were not able to go beyond 13 binary digits with the cross-validation trials because at that point we reached an implementation limit of the computer program. However, the predictions did form a partial pattern which enabled us to put an upper bound on p(0,0), as follows.

From what was said earlier, (1,1) predictions were not observed to occur. Therefore, all cases in class 1 were actually mis-classified, i.e. taking the cases in order of magnitude, every third prediction was of type (1,0). Sandwiched between these (1,0) predictions were others of type (0,0) and (0,1), occurring in twos. It was observed that two predictions of type (0,0) never occurred adjacent to one another, whereas two predictions of type (0,1) sometimes did - although this occurred with decreasing frequency as the number of bits increased. Now, if these observations hold true for any number of such binary attributes, this admittedly tenuous argument would suggest an upper bound of 1/3 for the probability of (0,0) predictions.

Thus it would seem that, for this example, we would never get more than one third (at most) of our predictions correct when using ID3, however many bits were used. Note that this is in marked contrast to what would happen if we just guessed "class 0" each time. Such a strategy would produce a probability of success of 2/3, without any computation at all!

Discussion

It is instructive to examine in detail what is happening inside ID3 when the induction process fails in this manner. In the induced classification tree that ID3 produces, the nodes consist of attributes and the branches consist of attribute values. The leaves are the predicted classes.

Because the branching of such a classification tree consists of a series of commutative processes, any given tree can be re-written, taking the attributes in a different order, without affecting the classification outcomes at all, although the resulting tree may have a different number of nodes.

If the classification tree for a particular incomplete data set is compared with the tree derived from the full data set - but rewritten to have the same branching order, then the two trees will be observed to have identical structures, except in the region of the omitted case. Obviously, the full tree classifies all cases perfectly. However, the incomplete tree fails to distinguish between the omitted case and the case that differs from it only in the last attribute branched on (in the order used). Thus the omitted case will be classified in the same direction as this similar case.

Now let us consider the possibilities that can arise. The class to which the similar case belongs determines the direction of prediction. If the similar case is in the non-divisble class, then the actual case may be in either class. However, if the similar case is in the divisible class, then an incorrect prediction is certain to occur because the number constituting the omitted case differs on just one attribute (i.e. by some power of two) from that constituting the similar case and hence cannot be divisble by 3. Thus all predictions for class 1 will be incorrect. This explains the absence of (1,1) events noted earlier. Of course, the asymptotic value for p(1,0) is 1/3. We have not yet found an analytical way of predicting the asymptotic values for p(0,0) and p(1,0) although, as suggested earlier, there is a tenuous argument for saying that the upper bound for p(0,0) is 1/3.

Having explained why prediction is so poor, it is now necessary to explain why the ID3 algorithm persists in deriving an induction tree when the attributes actually convey no real information about class membership. The reason lies in an inappropriate choice of *branching criterion* in ID3. When the classification tree is under construction, at each node a choice is made concerning which attribute to branch on next. ID3 uses an information-theoretic measure of tree complexity to select the next attribute, with the aim of reducing the complexity of the final tree. Unfortunately, this is not always the most appropriate procedure. For a problem such as that under consideration, this results in branching on attributes which convey no information whatsoever about class membership.

As a preferable alternative, we should consider branching criteria such as those suggested by Hunt et al (1966). Two possibilities, in particular, are worthy of mention. Firstly, transmitted information could be used as a measure of the extent to which attributes provide information concerning class membership. At each node, that attribute providing most information on class membership would be selected for branching. (Of course, it should be noted that, although this is an information-theoretic measure, it is not the same as that used in ID3 for assessing tree complexity).

The other idea is even more appropriate. This is to use the chi-square test as a measure of association between class and attribute. At each node, that attribute showing the strongest association with class is selected for branching, subject to the constraint that the value for X^2 must achieve statistical significance at some pre-determined level. The value of using a significance test as a stopping rule for the branching process is that with noisy data, chance frequency perturbations will be less likely to be seized on by the algorithm, taking the branching process farther than it should go. Of course, if branching is terminated before discrimination is complete, then the induction process is probabilistic. This topic and use of the chi-square test as a branching criterion is dealt with in more detail by White (1985). However, it is worth noting that, had this criterion been used in a probabilistic induction algorithm for the current example, then no branching at all would have taken place and the outcome would merely have been probabilities of class membership of 2/3 and 1/3 - which is entirely appropriate where the attributes convey no information at all about class membership. Indeed, not only is it considerably better than the result achieved by the minimal complexity criterion, it is the best that can be done.

Conclusions

What conclusions can be drawn from the peculiar behaviour of the ID3 algorithm in these circumstances? Firstly, the importance of cross-validation is re-affirmed. What appears to be perfect prediction when inside the training set can drop to considerably below chance level, with certain types of application, if the rules are applied outside the training set.

The second point that should be emphasised is that of the importance of the distinction between finite and infinite populations of cases. With the former, if the entire population of cases is used for deriving the rule, then there is no problem. No matter how complex the rule, the problem of over-fitting does not occur, simply because predictions are never made outside the training set. Such situations can arise in chess endgame analysis, for example. In contrast to this type of application, the domain of medical diagnosis is a typical example of one where we are dealing with an infinite population of cases and where the rules are applied to cases other than those used for training. In these circumstances, if the data are noisy, then it is probably a mistake to elaborate the rule at the training stage until all the exceptions are accounted for. This is overfitting.

Thirdly, this highly contrived example suggests the need to look more closely at the nature of induction algorithms, in order to determine whether they perform optimally under all conditions.

Fourthly, this example also highlights the need to examine more closely the relationship between statistical models and logical rules and the matter of whether the statistical approach is better for some types of problem and the logical approach for others. The simple answer to this is, of course, that the statistical approach is more appropriate when the data are noisy. However, this can be a facile answer if we bear in mind that we can reduce noise (either apparently or in reality) by using further attributes to distinguish between cases which clash in the original attribute space. If we are dealing with infinite populations, under what circumstances are we justified in removing the noise in this way (and using a logical approach) and in what circumstances would we be guilty of overfitting? In the logical approach, are there any characteristics of the induced classification tree that enable us to tell, or must we always employ the expensive technique of cross-validation? A possibility worth investigating is to examine the way in which the size (i.e. number of nodes of the induced tree) increases as the number of attributes is increased. It is noteworthy that in the experiment described here, the full induced trees had sizes which increased exponentially with number of attributes employed.

A related issue is the choice of branching criterion. If a finite population is involved and we have a complete data set, then the choice of branching criterion is not important as regards predictive accuracy and a minimal complexity tree may well be preferable on grounds of parsimony. On the other hand, if we are dealing with noisy data from an infinite population, it is necessary to ensure that branching is only carried out on attributes which are actually informative about class membership.

Finally, it is worth noting that this example provides a dramatic illustration of the crucial importance of choosing appropriate attributes. If we had used ternary digits instead of bits, the solution would have been trivial!

References

Hunt, E.B., Marin, J. and Stone, P.J. Experiments in induction. Academic Press, New York and London, 1966.

Michalsky, R.S. and Chilausky, R.L. Knowledge acquisition by encoding expert rules versus computer induction from examples: a case study involving soybean pathology. British Journal of Man-Machine Studies, (1980), **12**, 63-87.

Quinlan, J.R. Induction over large databases. Technical Report, University of Sydney, March 1979.

White, A.P. PREDICTOR: An alternative approach to uncertain inference in expert systems. In Joshi, A. (ed). Proceedings of the Ninth International Joint Conference on Artificial Intelligence, Los Angeles, 1985. Morgan Kaufmann, Los Altos, 1985.

P 140

BRITTLENESS AND MACHINE LEARNING

Stephen I. Gallant

February 22, 1986

College of Computer Science
Northeastern University
Boston, MA 02115 USA

P 141

ABSTRACT

Brittleness (or the lack of robust behavior) is a major problem in Artificial Intelligence, especially with respect to expert systems. A major cause of brittleness is tunnel vision, which is defined as making decisions based upon a few factors while ignoring other relevant factors.

This paper argues that tunnel vision and brittleness are fostered by some of the fundamental goals and models employed in Artificial Intelligence and that most common machine learning, knowledge representation, and expert system schemes are predisposed toward brittle behavior.

There are alternative techniques which avoid tunnel vision by taking into account all relevant factors. One such knowledge representation method employs networks of linear discriminants. This scheme is well suited for machine learning algorithms, for classification expert system knowledge bases, and, most importantly, for automatic generation of classification expert systems from training examples.

KEYWORDS: Knowledge Representation, Machine Learning, Expert System, Brittleness, Linear Discriminant, Tunnel Vision, MACIE

NOTE: Partially supported by a grant from the Northeastern University Research and Scholarship Development Fund. Thanks to Mark Frydenberg, Mitch Wand, Carole Hafner, Karl Lieberherr and Larry Bookman for helpful comments.

I. Introduction

On July 29, 1985, the launch of the Space Shuttle Challenger was partly aborted. Heat sensors on one of the three main engines indicated dangerously high temperatures which, if accurate, would have threatened the destruction of the craft and its human occupants. Computers immediately shut off the engine in question and the craft limped into orbit. Had this occurred 33 seconds earlier, the vehicle would have been forced to try a somewhat risky emergency landing in Spain.

Almost immediately mission controllers suspected that the engine was not at fault. They knew that heat sensors frequently failed and they also noticed that there were no other indications of engine failure, as would be expected in a true malfunction. Thus they quickly deduced that the problem was with the temperature sensors, not the main engine, and that the engine should not have been shut down.

Why was the program wrong and why were the controllers right?

The contrast is striking. The brittle rule--"If both heat sensors indicate overheating, shut off the engine"--is correct most of the time, but needlessly misses on some situations. The robust judgment of a human expert, on the other hand, is usually more reliable.

This situation dramatically illustrates the nature of brittleness, a troublesome problem in artificial intelligence models for knowledge representation, machine learning, and expert systems. When the input to a brittle system is slightly out of the ordinary, the system collapses. It doesn't bend, it breaks. We take this behavior as a loose definition of brittleness and also follow the common usage of robust to mean "not brittle".

II. Tunnel Vision

Returning to the Shuttle launch example, why was the rule brittle and the human judgment robust? The rule based its conclusion on exactly two factors, namely the two heat sensor readings. The controller examined all factors and acted differently. We may say that the rule suffered from tunnel vision because it examined only two key inputs, whereas the human expert took into consideration all relevant information.

One of the main points of this paper is that any system which significantly restricts the information entering into a decision is predisposed toward brittleness. In short,

TUNNEL VISION CAUSES BRITTLENESS.

Restating the situation in terms of mathematical functions, suppose we have a boolean decision function F based upon n input features (see Fig. 1). Often we can approximate F by a function F^* which examines many fewer features, yet correctly classifies inputs most of the time. However F^* will fail in exactly those cases where unexamined features are significant. To recapitulate, ignoring information may simplify decision functions, but it predisposes toward brittleness.

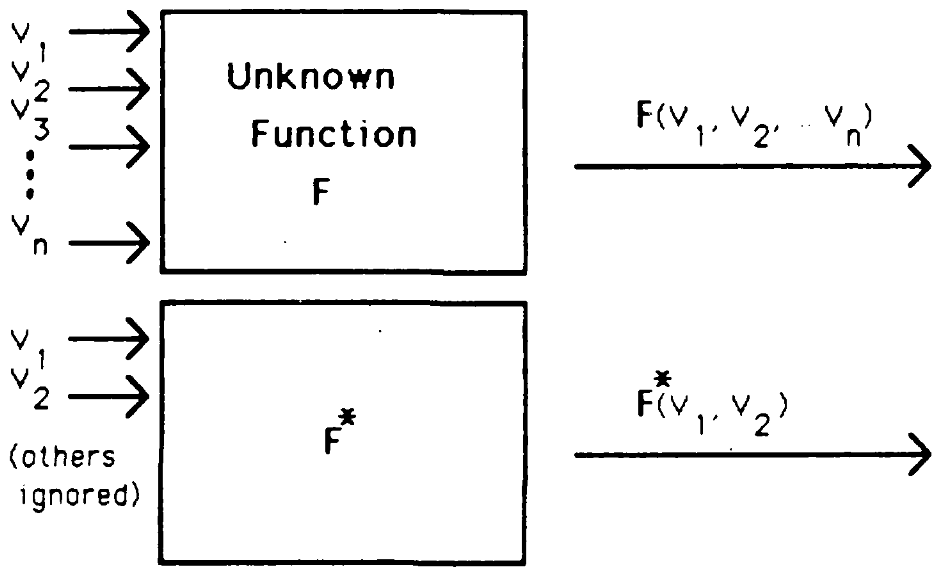


Figure 1 <u>Functional View</u> Function F* can approximate desired function F, but will be wrong for those cases where ignored inputs are necessary for correct output.

Perhaps the reader is due an apology at this point for having dwelt so long on such a simple point as "Tunnel Vision Causes Brittleness." However if this thesis is accepted, then the consequences are rather severe. It will be argued that _most of the knowledge representation, machine learning, and expert system generation schemes in current use_ suffer from (and often enthusiastically embrace!) aspects of tunnel vision and therefore _are predisposed toward brittleness_.

III. _Machine Learning and Knowledge Representation_

Let us examine machine learning and associated knowledge representation methods. One knowledge representation scheme frequently employed is conjunctive or disjunctive normal form logical expressions (IF-THEN rules) that are as _simple_ as possible.

A disjunctive normal form (DNF) expression is generated by the OR'ing together of terms, where each term consists of AND'ed boolean factors. NOT's may now be sprinkled in to taste. E.g.:

$$C = (F1 \text{ and } F5 \text{ and not } F9) \text{ or } (F3 \text{ and not } F1).$$

Simplicity is assumed beneficial because of ease of comprehension by humans and ease of communication. Humans, it is argued, often use such expressions when explaining their own decisions.

But this is not a plausible justification for using simple DNF rules. Human communication is a very special activity. By necessity, reading transpires slowly and in a more or less sequential fashion.

Speech and writing are even slower and more sequential. These are constraints imposed upon human communication by the communication media which are not present for other mental activity. It would therefore be rather surprising if the knowledge representation system employed in the constrained communication environment were also employed in the unconstrained (or at least differently constrained) mental environment. As a computer analogy, the CPU does not communicate with itself and memory by first translating everything into ASCII!

Thus the fact that humans give rule-like explanations gives very little support to the use of rules or logical expressions for knowledge representation.

A substantial amount of effort has been devoted to machine learning algorithms that produce short and simple rules. This work is mathematically interesting and useful for VLSI design and other areas. However the simpler the logical expression, the more it tends to suffer from tunnel vision. Consider the following example.

	Factors										Correct Response
	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	
Training Example 1	T	T	T	T	T	T	T	T	T	T	T
Training Example 2	F	F	F	F	F	F	F	F	F	F	F

Figure 2: Machine Learning Example.

In Figure 2 we have two training examples for a classification problem based upon 10 factors. Most machine learning methods would produce a beautifully simple rule for the example such as:

```
If $F_1$ is TRUE then the classification is TRUE
                 else the classification is FALSE.
```

Yet this rule is incorrect!

The rule is incorrect because it does not agree with human common sense for a large number of possible inputs. For example, consider the following input:

$$F_1 = \text{True}; \quad F_2 = F_3 = \ldots = F_{10} = \text{False}.$$

The simple rule above gives what must be considered the wrong answer due to its tunnel vision.

A correct, robust rule is given in Figure 3.

```
If the majority of known factors are TRUE
          then the classification is TRUE

Else if the majority of known factors are FALSE
          then the classification is FALSE

Else the classification is chosen
          to be TRUE or FALSE arbitrarily.
```

Figure 3: Correct, robust rule.

The correct rule works much better in noisy environments. For example, suppose each F_i is only 85% reliable. Then the original rule is wrong 15% of the time, while the new rule is wrong less than 0.6% of the time. In general it is preferable to take all relevant features into account, including redundant features, when dealing with noisy environments. Otherwise tunnel vision results and decisions are less reliable.

Another advantage of the correct rule is its applicability to situations where only partial information is available. For example suppose a subset of F_2-F_{10} is known but F_1 is unknown. The rule in Figure 3 is still corect while the original rule is useless since it only examines F_1.

These examples illustrate the drawbacks of purchasing simplicity at the expense of tunnel vision, particularly if noisy systems are involved.

Machine learning methods are not the only problem; there is a severe problem with the knowledge representation schemes used in conjunction with machine learning. The reader may have already objected to the language in the correct rule above, in particular the use of the phrase "majority of the known factors." For example, what if we change the problem by renaming every even numbered factor to its opposite value as in Figure 4.

	Factors										Correct Response
	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	
Training Example 1	T	F	T	F	T	F	T	F	T	F	T
Training Example 2	F	T	F	T	F	T	F	T	F	T	F

Figure 4: Machine Learning Example with even factors renamed.

Conceptually the situation has not changed at all -- we have only relabelled factors -- yet stating a correct rule has suddenly become messy. Thus the inherent simplicity of a correct rule can be lost when representing it as a logical expression, and it may be difficult to describe such rules informally.

An even more serious problem is that ${}_{10}C_5$ (combinations of 10 objects chosen 5 at a time) terms are needed to represent the rule in Figure 3 using disjunctive normal form expressions. The number of terms in a disjunctive normal form expression can quickly get out of bounds for slightly larger problems, since combinations grow exponentially. Therefore disjunctive normal form expressions cannot represent rules of the form "K or more out of L factors are True" for even moderately sized problems due to the number of terms required. This situation can be ameliorated somewhat by generating new variables which correspond to intermediate terms, but few (if any) machine learning schemes do this.

The above arguments lead us to conclude that simple logical expressions, as a means of knowledge representation, are not well suited for expressing robust rules.

Another common knowledge representation system frequently used with machine learning is the decision tree. To demonstrate the decision tree's unsuitability for robust rules, the reader need only construct the tree for the correct rule given in figure 3 where each decision variable is a single factor's value (e.g. F_3 = TRUE, FALSE, or UNKNOWN).

Again we can merge some of the nodes to cut down the dimensionality of this problem, but it is not clear how to do this in general and there are still too many nodes.

Frame based knowledge representation systems, by the same arguments, are also predisposed toward brittleness. "Slots" are clearly the handiwork of the Tunnel Vision Demon. In the classical frame setup, there cannot be too many slots since all must be satisfied if the corresponding object is to be recognized. Yet limiting the number of slots produces tunnel vision and brittle representations. See Brachman[1985] for an interesting discussion of these problems.

IV. Expert Systems

Expert systems are built upon quicksand. Most employ a knowledge base consisting of logical expressions, predisposing them toward tunnel vision. Even worse, the process of constructing such a knowledge base reinforces this tendency.

It is only natural for a knowledge engineer to seek out simple rules for decision variables. This means finding a small set of factors which determine the situation most of the time, as in the Space Shuttle heat sensor rule. It is hard enough to build and debug a knowledge base consisting of simple rules; it is even more difficult if complex rules are used. Yet simple rules usually result in tunnel vision and brittleness.

It is difficult for a knowledge engineer to escape this dilemma by using complex, robust rules. Suppose, for example, a less brittle engine shut-off rule is desired for the Space Shuttle. One possible approach is to start with the original rule which is based upon two sensors and modify it by considering the *influence of other factors*, taking them one by one. Generally many other factors must be considered to avoid brittleness. Great care must be taken that each additional factor does not increase the current set of rules by some fixed fraction, or else the number of rules will grow exponentially. But if the number of rules is kept small, individual rules will increase in complexity and become increasingly difficult to debug.

Even with a suitable knowledge representation scheme, it may not be humanly possible to hand-build a robust knowledge base for a large system. It appears that machine learning techniques are <u>necessary</u> (though perhaps not sufficient) for constructing a large, robust expert system.

But what are the learning techniques and where is the knowledge representation scheme powerful enough to express robust rules?

IV. <u>Is There No Hope</u>?

Lest the reader become despondent and do something drastic such as take up assembly language programming, we hasten to point out that not all schemes for knowledge representation, machine learning, and expert system generation suffer from tunnel vision.

Perhaps the simplest way around tunnel vision is to remember all training examples. When a new situation is presented, the retained example closest to that situation is chosen for emulation. There is an expert system shell currently being sold which uses just this procedure. Among the drawbacks to this scheme are the fact that storage requirements and response times grow with the number of examples

A second scheme which is more to our liking uses Linear Discriminant Networks (LDN's) for knowledge representation. (For a good general reference on linear discriminants see Duda & Hart [1973]. We will briefly show how some of the previously discussed examples fit into this framework.

Let us reconsider the machine learning example from Figure 2. Since all factors are boolean (or unknown), it is convenient to represent their values numerically as follows:

$$F_i = \begin{Bmatrix} +1 \\ -1 \\ 0 \end{Bmatrix} \quad \text{if} \quad F_i \quad \text{is} \begin{cases} \text{TRUE} \\ \text{FALSE} \\ \text{UNKNOWN} \end{cases}$$

We can now represent a linear discriminant decision rule by a vector of integer weights:

$$< W_0 \mid W_1, W_2, \ldots, W_n >$$

where the rule is computed by comparing the sum of weighted factors to 0:

$$\text{If } W_0 - \sum_{i=1}^{n} W_i F_i \left\{ \begin{matrix} >0 \\ <0 \\ =0 \end{matrix} \right\} \text{ then conclude } \left\{ \begin{matrix} \text{TRUE} \\ \text{FALSE} \\ \text{UNKNOWN.} \end{matrix} \right.$$

Thus the robust rule given in Figure 3 could be represented:

$$< 0 \mid 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 >$$

It is now trivial to represent a robust rule for the modified example in Figure 4:

$$< 0 \mid 1, -1, 1, -1, 1, -1, 1, -1, 1, -1 >.$$

Any logical expression can be represented by a set of linear discriminants as demonstrated in Figure 5. Thus a network of linear discriminants can represent any function.

Disjunctive Normal Form Expression:

F = (A and not B and C) or
(A and D) or
(B and not D and not E)

Corresponding Linear Discriminant Network:

		Inputs									Outputs
	W_0	W_A	W_B	W_C	W_D	W_E	W_{D1}	W_{D2}	W_{D3}		
Discrim. #1:	< -2	1	-1	1	0	0	0	0	0	>	--> D1
Discrim. #2.	< -1	1	0	0	1	0	0	0	0	>	--> D2
Discrim. #3:	< -2	0	1	0	-1	-1	0	0	0	>	--> D3
Discrim. #4:	< 2	0	0	0	0	0	1	1	1	>	--> F

Figure 5: Disjunctive Normal Form Expression represented as Linear Discriminant Network. All inputs and outputs are +1 (True) or -1 (False). Each term in the DNF expression is computed by a single discriminant, and the last discriminant OR's the terms to compute F.

Methods for computing discriminant weights (and the network) are currently being studied by a number of researchers: Hinton [1984], Barto[1985], Gallant [1986a,b], and Rumelhart[1985].

The MACIE system for generating expert systems from training examples uses some of these techniques. A linear discriminant network is first computed from training examples. Then a special inference engine uses the LDN as a knowledge base to perform inferences, identify useful unknown information, compute likelihoods, and give justifications for inferences (in the form of simple IF-THEN rules). The Appendix gives a sample run of MACIE that demonstrates its explanation facility. For a more complete description see [Gallant 1985a].

How would the Shuttle rule for engine shut-off look using LDN's? One simple rule might say

"If heat sensors 1 and 2 (represented by S_1 and S_2) fire and 5 out of 100 other factors (represented by F_1-F_{100}) give positive indications, then shut down the engine."

Here's the discriminant network (with only one discriminant) that expresses this rule:

$$\begin{array}{ccccccccc} C & S_1 & S_2 & F_1 & F_2 & F_3 & \cdots & F_{100} \\ < -109 \,\vdots & 100, & 100, & 1, & 1, & 1, & \ldots, & 1 > \end{array}$$

So if both sensors fire along with F_1 through F_5, the calculation gives:

$$-109 + (100)(+1) + (100)(+1) + 5(1)(+1) + 95(1)(-1) = 1 > 0$$

so the engine shuts down.

Of course we may want to make some of the 100 factors more important than others or have shutdown if only 1 sensor indicates overheating when accompanied by lots of other lessor indications. This can be done by adjusting weights appropriately or by adding discriminants to the network when necessary.

Procedures for computing LDN's are described in the previously mentioned references. Current research seeks to determine whether LDN generation methods are fast enough to be practical for larger problems. (MACIE style expert systems are very fast at "run time", once an LDN has been generated at "compile time".)

V. Summary and Conclusion

We have defined tunnel vision as the focusing on a few variables while ignoring other significant variables and have claimed that tunnel vision predisposes a system toward brittleness. Based upon this criterion we have found:

1. Commonly used knowledge representation schemes such as logical expressions, decision trees, and frames are not well suited for representing robust systems.

2. Machine learning techniques producing rules which minimize the number of factors examined are interesting mathematically but fail to address problems of brittleness and are not well suited for noisy, redundant environments.

3. It is extremely difficult to hand-build a large expert system that is robust. Machine learning techniques are probably necessary, though perhaps not sufficient, for such a task.

4. Some knowledge representation, machine learning, and expert system schemes do not encourage tunnel vision. In particular, those methods based on linear discriminant networks seem promising.

We conclude that additional effort should be applied to knowledge representation and machine learning schemes that are more conducive to robust systems. Finally, we reemphasize the necessity of employing machine learning techniques for the construction of large, robust expert systems.

P 157

REFERENCES

[Barto 1985]
Barto, A. F. Learning by Statistical Cooperation of Self-Interested Neuron-like Computing Elements. Human Neurophysiology, to appear.

[Brachman 1975]
Brachman, R. J. "I Lied about the Trees" Or, Defaults and Definitions in Knowledge Representation. AI Magazine, Fall 1985, 80-93.

[Duda & Hart 1973]
Duda, R. O. & Hart, P. E. Pattern Classification and Scene Analysis. (1973) John Wiley & Sons, New York.

[Gallant 1985a]
Gallant, S. I. Automatic Generation of Expert Systems From Examples. Proceedings of Second International Conference on Artificial Intelligence Applications, sponsored by IEEE Computer Society, Miami Beach, Florida, Dec. 11-13, 1985.

[Gallant 1986a]
Gallant, S. I. Optimal Linear Discriminants. Technical Report SG-85-30, Northeastern University College of Computer Science. To appear, Eighth International Conference on Pattern Recognition, Paris, France, Oct. 28-31, 1986.

[Gallant 1986b]
Gallant, S. I. Three Constructive Algorithms for Network Learning. To appear: Eighth Annual Conference of the Cognitive Science Society, Amherst, Ma., Aug. 15-17, 1986

[Hinton 1984]
Hinton, G. E., Sejnowski, T. J., & Ackley, D. H. Boltzmann Machines: Constraint Satisfaction Networks that Learn. Technical Report CMU-CS-84-119, Carnegie-Mellon University Department of Computer Science

[Rumelhart 1985]
Rumelhart, D. E., Hinton, G. E., & Williams, R. J. Learning Internal Representations by Error Propagation. ICS Report 8506, Institute for Cognitive Science, University of California, San Diego, Sept. 1985.

APPENDIX

APPENDIX: EXAMPLE OF AUTOMATIC GENERATION OF AN EXPERT SYSTEM FOR DIAGNOSIS AND TREATMENT OF PLETHORAL DISEASE

Note: Input to learning program given in figure 2. Learning matrix produced is given in figure 3. File has been edited slightly for readability.

```
MACIE Version 1.0   (c) 1985 S. I. Gallant

Enter name of file containing Knowledge Base.

mtest

Enter initial values for T. I. or G variables.
Format:  Variable number. value. ...
Uninitialized variables are set to 0 (UNKNOWN)

Numbers and names of variables:
           1:  Fever between 100 and 102 F
           2:  Normal Weight
           3:  High White Cell count
           4:  Toughening of skin
           5:  Reddish tinge to hair
           6:  Allergic to Cacamycin
           7:  Inflamed Plethorus
           8:  Poblanocosis
           9:  Administer Cacamycin
          10:  Administer Merdecillin

1t 3f 5f

CONCLUDE:  (8) Poblanocosis FALSE.

PURSUING Administer Merdecillin
PURSUING Administer Cacamycin

Is the patient allergic to Cacamycin?  (y/n/?)

?

Enter 'q' for explanation of the last question
      'v #' for explanation of variable number #

v8 (Why was poblanocosis determined to be FALSE?)

Poblanocosis is FALSE due to the following rule:

    IF High White Cell count is FALSE
       AND Reddish tinge to hair is FALSE

    THEN CONCLUDE Poblanocosis is FALSE.

Is the patient allergic to Cacamycin?  (y/n/?)

?

Enter 'q' for explanation of the last question
      'v #' for explanation of variable number #

q (Why was this question asked?)

Allergic to Cacamycin is a factor
   which gives negative support for
        Administer Cacamycin
   which gives negative support for
        Administer Merdecillin
   which is the Goal Variable with highest
        Likelihood.

PURSUING Administer Merdecillin
PURSUING Administer Cacamycin

Is the patient allergic to Cacamycin?  (y/n/?)

y

CONCLUDE:  (9) Administer Cacamycin FALSE.

PURSUING Administer Merdecillin
PURSUING Inflamed Plethorus

Is the patient's weight normal?  (y/n/?)

?

Enter 'q' for explanation of the last question
      'v #' for explanation of variable number #

v9

Administer Cacamycin is FALSE due to the
following rule:

    IF Allergic to Cacamycin is TRUE

    THEN CONCLUDE Administer Cacamycin is FALSE.

Is the patient's weight normal?  (y/n/?)

n

CONCLUDE:  (7) Inflamed Plethorus TRUE.
CONCLUDE:  (10) Administer Merdecillin TRUE
```

(from GALLANT 1985a)

PROTOS: An Experiment in Knowledge Acquistion for Heuristic Classification Tasks[1]

Bruce W. Porter
E. Ray Bareiss
Computer Sciences Department
University of Texas at Austin

Abstract

PROTOS is an experiment in acquiring, applying and revising expert knowledge for heuristic classification. Learned knowledge includes exemplar-based categories and domain knowledge to support explanation and pattern matching. Research on PROTOS has revealed fundamental issues in concept formation and classification concerning *representation* of ill-defined, "fuzzy" categories, multiple applications of learned knowledge, and the context in which learning takes place. This paper highlights these issues, describes the PROTOS approach to learning heuristic classification, and surveys related research.

1. Introduction

Machine learning research on concept formation has ignored many thorny issues. These issues fall into three categories:

1) learning natural, "fuzzy" concepts. Natural concepts describe our everyday world. Because of the significant variability of objects in our world, examples of natural concepts are rarely *all or none* [LANG84]. Classical definitions, which consist of necessary and sufficient features, do not exist for natural concepts [SMIT81, WITT53]. Much of the research in concept formation has focussed on learning artificial concepts, which *are* classically defined, such as "arch" [WINS75], "prime number" [LENA77], and "Good Problem-States for Applying Integration by Parts" [MITC83, PORT86]. Learning natural concepts requires a radical departure from past induction-based learning strategies.

2) multiple applications of learned concepts. The knowledge acquired by concept formation programs is sparce. This is apparent when you consider the range of functions that learned knowledge should support. This includes classification of new objects, summarization of training instances, generation of examples of a concept, prediction of unseen features of a new object, interpretation of "fuzzy" examples, and explanation. Concept formation research has addressed only the first two functions. New objects are identified by matching with a generalized concept description, which is acquired by

[1] Support for this research was provided by the National Science Foundation under grant number IST-8510999 and the Army Research Office under grant number ARO DAAG29-84-K-0060.

summarizing the training set via induction. Supporting the other important applications of learned knowledge requires de-emphasizing the role of generalization in concept formation.

3) the context in which the learning takes place. Concept formation from either unclassified or classified training examples is an artificially difficult cognitive task. Theoretical analysis suggests that large classes of disjunctive concepts may not be learnable from this training [VALI84]. Psychological evidence suggests the importance of a responsive environment for concept learning (*e.g.* [HILG75]). Experimentation, problem solving, and instruction are crucial to learning. Learning is in support of problem solving. Instruction is provided by a teacher who explains the reasons for problem solving failures. These important roles of the teacher and the problem solver in concept formation demand reappraising our machine learning models.

The goal of our research is to address these issues in a computational model of the acquisition, use, and revision of natural concepts. We are motivated by discoveries in cognitive science concerning concept formation and representation (*e.g.* [SMIT81, ROSC78, NELS74]) but the focus of our research is the design of a useful architecture for the acquisition and application of knowledge by computer. We are constructing a learning system, named PROTOS, which evolves into an expert system for a complex task. PROTOS learns by attempting to solve problems posed by a domain expert; focussed interaction with the expert uncovers the cause of problem solving failures and guides learning.

The problem solving task for PROTOS is classification. The vast majority of expert systems are designed for this task. As characterized by Clancey [CLAN85], these systems have an inference structure that relates an object description to a pre-enumerated set of classifications. Since the classification of an object is typically based on an imperfect match, heuristics are used to bolster the match. Clancey defines *heuristic classification* to be the use of non-hierarchical, uncertain inference in classification. PROTOS is an architecture for supporting the acquisition and application of domain knowledge for heuristic classification.

PROTOS learns by attempting to classify objects. Classifying an object entails recalling a similar object which has been previously classified and constructing a convincing explanation of the similarity. If PROTOS fails to correctly classify and explain then the teacher intervenes. The teacher might correct the classification or provide additional explanation. The role of PROTOS as a student is to evaluate explanations and to maintain memory organization.

The remainder of this paper describes the motivation and design of PROTOS. Related research in machine learning is presented in the next section. Section 3 incrementally develops the PROTOS approach to learning heuristic classification knowledge. Finally, section 4 outlines our current and future research.

2. Related Work

Machine learning is of central importance in the recent surge of interest in knowledge acquisition tools for expert system. Most of this research focusses on the interaction with an expert to elicit heuristic classification knowledge. Knowledge representation and application issues are not addressed because the goal is to produce rule-based expert systems, which are well understood. The research is pragmatic, borrowing little from cognitive science. This section reviews this related work by describing the viability of each approach for eliciting knowledge from an expert.

Rule induction algorithms (*e.g.* ID3[QUIN86], Plant/DS[MICH80]) have a limited range of applicability. Typically, these algorithms are given training in the form of classified examples from a teacher. Similarities in the training examples suggest generalizations which are represented as rules. The teacher is "outside the learning loop" so there is no chance to guide the search for useful generalizations. The algorithms are highly sensitive to *a priori* bias and peculiarities in the training set. Most of the algorithms are non-incremental since they require all training examples before learning begins. The most serious limitation of these algorithms is that the learned rules only support the function of classification. The knowledge base formed by similarity-based inductive learning is unsuitable for other essential functions such as explanation [SCHA86].

Another approach to learning heuristic classification knowledge is to automate the activities of a knowledge engineer interviewing a domain expert. Examples of this approach include MORE [KAHN85], Tieresias[DAVI77], ROGET[BENN85], and the Expertise Transfer System [BOOS84]. The main activities of these systems include analyzing the structure of the knowledge base, detecting possible weaknesses and inconsistencies, and focusing the expert on corrections. Unlike rule induction, the learning is incremental and guided by the domain expert. The sources of power in most of these systems are the connection with an expert system guided by (partial) domain knowledge and the ability to ask focused questions of the expert. An analysis of the knowledge can reveal structural problems of known types, which are described to the expert for correction. This approach to acquiring heuristic classification knowledge demonstrates the viability of knowledge-base analysis and focussed interaction with the teacher. However, the limitations of rule-based systems impair the approach. In particular, these techniques require a richer language for knowledge representation than rules provide.

Learning apprentice systems are a promising approach to knowledge acquisition. As defined by Mitchell, *et.al.* [MITC86], a *learning apprentice system* is:

"... a knowledge-based system that provides *interactive* aid in solving some problem, and that acquires new domain knowledge by generalizing from training examples acquired through the *normal course* of its use."

These systems have been demonstrated in various domains including VLSI circuit design [MITC85, MAHA85, ELLM85], game playing [MINT84, FLAN86], and story understanding [DEJO81]. Learning apprentice systems observe the expert's performance and construct an explanation of unexpected actions. Each explanation is constructed from *a priori* domain knowledge, generalized, and saved for future use. The source of power of these systems is the domain knowledge for constructing explanations. Acquiring and representing this initial knowledge is a major obstacle to their construction. PROTOS can be viewed as an apprentice system which learns the domain knowledge required for explanation-based reasoning.

3. The PROTOS Approach to Learning and Problem Solving

The primary tenet of PROTOS is that classification of objects must be *model-directed*. A model guides classification by providing biases and expectations. Classification of an object is the process of selecting an appropriate model and coercing the description of the object to conform to the model. Learning to do this task requires learning and indexing models and acquiring the domain knowledge necessary to coerce object descriptions.

This section incrementally develops the PROTOS approach to learning and problem solving for heuristic classification tasks. First, we review the seductive appeal of inductive learning and the serious consequences of adopting a representation of categories based on induced features. An *exemplar-based* category representation, which de-emphasizes the role of induction, is presented as the foundation of PROTOS. Second, we describe PROTOS1 which learns and uses *explanations* to both perform model-directed classification and unite category members. Finally, we discuss PROTOS2 which learns and uses differential *reminding power* of object features to guide the search for an appropriate model.

3.1 Exemplar-Based Categories: The Foundation of PROTOS

Research on concept formation usually makes the simplifying assumption that a concept can be represented by a *classical definition*. A classical concept definition is a set of necessary and sufficient conditions for an object to be an instance of the concept. This section explains the strong appeal of this assumption for AI researchers and the serious consequences of its adoption. Exemplar-based categories are presented as a promising alternative.

There are distinct benefits from assuming a classical definition of concepts. The first is that concept formation is reduced to induction. Given a collection of positive and negative examples of a concept, necessary features are extracted from near misses, and sufficient features are extracted from positive examples. Concept formation is search with standard generalization and specialization operators [MITC82, MICH83]. The defining features of the concept, shared by all positive examples, are inferred by induction over the training set.

The second benefit is that object identification is reduced to deduction. One of the primary uses of learned concepts is to identify an object from a partial description. In this sense concepts are pattern recognizers. Object identification simply requires a successful match of the object with the defining features of the concept. The identification is all or none; unclear classifications are not considered because of the restrictive nature of classical concept definitions.

Unfortunately, classical concept definitions *only* work in artificial domains, such as those frequently used to demonstrate AI systems. There are four major shortcomings of classical concept definitions for natural concepts (see [SMIT81, BARE86] for additional discussion). First, the defining features of most natural concepts cannot be enumerated. This problem is easily observed by trying to form classical definitions for natural concepts such as *chair, bird* or *appendicitus condition*. Second, classification of some objects is unclear. For example, tomatoes share features of both vegetables and fruits. Third, many concepts are disjunctive. A vehicle can have either a steering wheel or handlebars. Forth, there are variations in typicality among instances of a concept. Some instances are "better" exemplars of the concept than others.

One solution to the problems of classical category definitions is to weaken the requirements for category membership. Rather than attempting to describe category members using necessary and sufficient features, a *probabilistic representation* uses weighted features and a threshold for identification. For example, a *vehicle* might be represented as:

$$\langle engine(.5), steeringwheel(.3), peddles(.4), handlebars(.2)\rangle$$

$$\text{matching threshold} = .9$$

Probabilistic definitions support object identification without requiring a perfect match. An identification succeeds if the combined weights of the matched features exceeds the identification threshold. This form of representation is commonly used in expert systems, such as MYCIN[SHOR76] and Internist[POPL82].

The probabilistic representation [SMIT81] avoids the problems of the classical representa.ion but introduces new problems of its own. Information about correlations among features is lost. For example,

- *engine* co-occurs with *steeringwheel* and with *handlebars*.
- *steeringwheel* never co-occurs with *handlebars*.
- *peddles* co-occurs with *handlebars* but rarely with *steeringwheel*.

Without this information, realizable instances of a concept cannot be generated, and unobserved features of an identified instance cannot be predicted. In summary, the probabilistic representation can encode fuzzy categories, but the fuzziness cannot be controlled.

The *exemplar representation* of categories [SMIT81] is a partial solution to the problems of classical and probabilistic representations. Rather than attempting to form a conceptual abstraction of the descriptions of category members, this representation is extensional. A category is simply a collection of its typical and atypical exemplars.

Unlike "compiled" knowledge representations, an exemplar-based category structure can support multiple functions. **Classification** of a new object is performed by finding the exemplar(s) in the knowledge base which matches it most closely and assigning the new object to the same category. **Explanation** of a classification is facilitated by reporting a similar, known exemplar. **Prediction** of unseen features of an object is based on feature correlations in the closest matching exemplar(s). **Example generation** simply involves exemplar retrieval (perhaps ordered by typicality).

Despite these advantages, the exemplar-based representation of categories introduces some problems. First, exemplar-based categories lack cohesiveness [SMIT81]. The underlying commonality of the category members is not explicit in the representation. As described in the next section, PROTOS1 extends the exemplar representation to address this problem; category members are united by explicit explanations of the rationales behind their classification. Second, an object to be identified may not exactly match a stored exemplar. This problem is addressed in PROTOS1 by *translating* object features to bolster a partial match. Third, object identification requires finding a "similar" exemplar. This search for candidates must be guided. As described in section 3.3, PROTOS2 guides the search with learned knowledge of the *reminding power* of features with respect to categories.

3.2 PROTOS1: Explanations Support Inexact Identification

PROTOS1 is an implemented system which demonstrates *knowledge-based pattern matching* to support the identification task. Knowledge-based pattern matching is motivated by the observation that learned categories guide our interpretation of new objects. An object *O* is identified by recalling a learned exemplar which shares the salient features of *O* and coercing matches among the remaining features. The recalled exemplar serves as a model; effort is expended to confirm the expectations generated by the model. PROTOS1 learns and uses exemplars as models for guiding identification; PROTOS2 indexes exemplars according to their appropriate uses as models and learns how much effort to expend on knowledge-based pattern matching.

Knowledge-based pattern matching is similar to the matching procedure in explanation-based learning (*e.g.* [MITC86, DEJO81]) and constructive induction [STEP86]. Each employs domain knowledge to construct an explanation of the relationships between observed, extrinsic features and criterial, intrinsic features. However, PROTOS1 *learns* the requisite knowledge and provides a richer language for explanations.

Domain knowledge to enable knowledge-based pattern matching is learned from explanations of classified objects presented by the teacher. This differs

```
Given object O to be classified
REPEAT
  classify:
    Use domain knowledge to explain the similarity of O to a
    previously classified object (exemplar).
    Analyse the explanation.
  learn:
    IF the classification and explanation are correct
    THEN
      IF object and exemplar are identical
      THEN Increase confidence in the exemplar
      as a good representative of the category.
      ELSE Add O as a new exemplar of the category.
    IF the classification is incorrect or the explanation is weak
    THEN Solicit domain knowledge from the teacher.
UNTIL the teacher is satisfied.
```

Figure 1
Overview of PROTOS1

from the training provided to most concept formation systems. For example, ID3[QUIN86] and Plant/DS[MICH80] are given a featural description of an object (*i.e.* the "input" to an expert) and the object's classification (*i.e.* the "output" from the expert). This training is relatively easy to provide and is well-suited to inductive learning but is too sparce. Information is not provided concerning how the features of an object relate to its classification (*i.e.* the "intermediate states" of the expert's reasoning).

Explanations are provided to PROTOS1 by the teacher when independent problem solving fails. The failure might be due to an incorrect object classification or a "buggy" explanation for a classification. Given a description of an object to be classified, PROTOS1 first attempts classification by constructing an explanation and then learns from the result. The abstract algorithm in figure 1 describes this two step process.

Knowledge-based pattern matching relies on constructing explanations of similarity. PROTOS1 provides a more expressive language for explanation than the commonly used language of production rules (*e.g.* [DAVI77, CLAN83, MITC85]). The "right arrow" which relates antecedent to consequent in a production rule is ambiguous. In disambiguating the "right arrow" relation, we have identified six categories of relations:

1) feature to definition mappings. Common sense or a domain fact relates two features. For example, adolescents are minors.

2) structural to functional feature mappings. A perceived feature implies or enables the function of an object. For example, wings enable flight.

3) circumstantial to inferred feature mappings. A perceived feature suggests a potentially more relevant feature. For example, fangs implies carnivorous.
4) specialization to generalization mappings. This is the traditional taxonomic relation. For example, beagle is a specialization of dog.
5) current state to successor state mappings. An underlying mechanism causes some change over time. For example, air pollution causes acid rain.
6) part to whole mappings. A feature is a part of an assembly. For example, wheels are part of a car.

The explanation language in PROTOS1 is an elaboration of these broad categories. PROTOS1 knows about these relations from *a priori* knowledge and from past examples of their use in the domain.

Training provided to PROTOS by the teacher is a featural description of a training example augmented with an explanation of the relevance of each feature to the classification of the example. The explanation of the relevance of a feature is a path through domain knowledge which justifies the significance of the feature. For example, *V8engine* is a feature of an exemplar of the category *car*. One explanation associated with the feature is:

$$V8engine \xrightarrow{spec} engine \xrightarrow{enables} movement \xrightarrow{func} vehicle \xrightarrow{genl} car$$

A network of domain knowledge, called a *category structure*, is incrementally built up from the exemplars and explanations provided during training. Nodes in the network are exemplar-based categories, as described in the previous section. Arcs in the network are relations from the explanations of training examples. For example, the category structure containing the preceeding explanation is shown in figure 2.

An important representational capability is the elaboration of features to arbitrary levels of detail. In figure 2, the object feature *V8engine* is itself an exemplar-based category. This ability to examine a feature "under a microscope" has two important advantages for PROTOS1:

1) The teacher can shift the level of granularity of the description of training examples. After teaching PROTOS1 an exemplar-based category, the teacher can use the category as a higher-level feature.
2) PROTOS1 can bolster a suspected high-level, featural match by attempting a more detailed match with exemplars of the features.

A priori knowledge in PROTOS1 is used to critique explanations generated from either knowledge-based pattern matching or teacher training. For example, a long explanation path containing circumstantial associations is more suspect than an identical match. PROTOS1 evaluates each explanation using criteria such as path length, types of relations used, supporting explanations, and competing

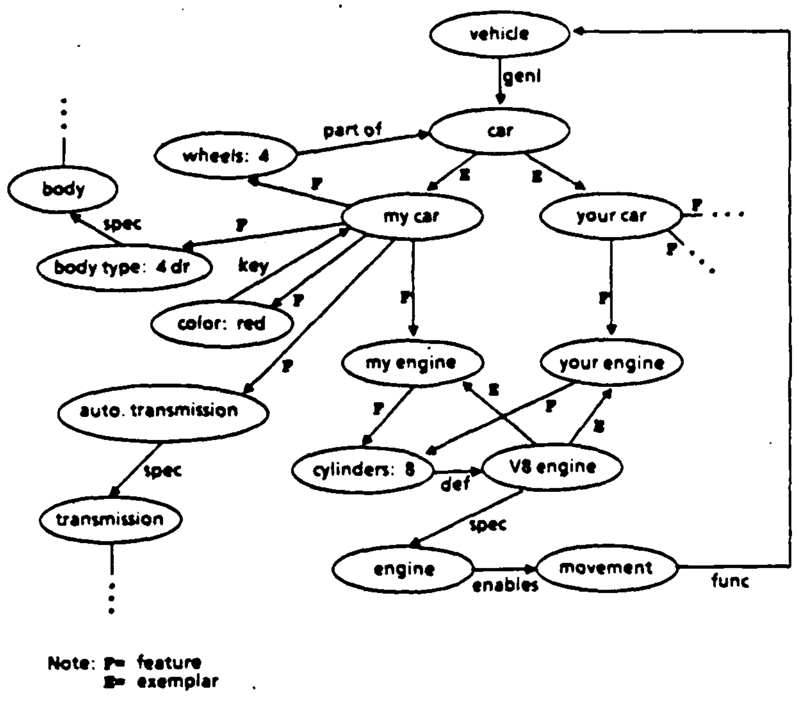


Figure 2
A Sample Category Structure

classifications (*c.f.* [COHE85]). PROTOS1 solicits knowledge from the teacher to clarify and bolster a suspicious explanation. Because explanations refer to specific exemplars, discussion with the teacher is highly focussed [PORT85].

Knowledge-based pattern matching introduces critical problems of efficiency which are not addressed by research in explanation-based learning. In particular, when attempting to classify an object by matching it with a similar exemplar, two issues arise:

1) Because of the computational expense of knowledge-based pattern matching, heuristic knowledge is required to limit the pattern matching operations which are attempted. This knowledge guides the selection of promising match candidates and determines how much effort to expend on confirming the match. How is this knowledge acquired and used?

2) Once the object is classified, it must be integrated with neighboring exemplars. This involves adding the object as an exemplar or generalizing with very similar exemplars. How is the set of "neighboring" exemplars determined?

3.3 PROTOS2: Remindings Support Efficient Indexing

We are currently implementing PROTOS2 which addresses the two problems identified in PROTOS1. The primary addition to the system is learning and using differential *reminding power* of object features. The features of an object to be identified can provide clues to which classifications to attempt. These clues suggest initial models which guide knowledge-based pattern matching. This indexing efficiency for object information is part of the general issue of reminding discussed by [SCHA82, ROSC75].

Object features can remind PROTOS2 of past exemplars or learned categories. For example, the feature *engine* has reminding power for the category of *car* while the feature $leaky - trunk$ has reminding power for the exemplar $my - car$. *Engine* is shared by all exemplars of *car* while $leaky - trunk$ is idiosyncratic of $my - car$. In general, the reminding power of a feature is strongest for the category or exemplar which has the highest conditional probability given the feature.

PROTOS2 learns the reminding power of features by both similarity-based and explanation-based analysis. The similarities in multiple examples and the explanations of individual examples determine the reminding power of features. PROTOS2 might receive many exemplars of *car* which have an *engine* or a single exemplar, $my - car$, with the explanation of the feature:

$$engine \xrightarrow{partof} car \xrightarrow{exemplar} my - car$$

Both suggest that *engine* has useful reminding power at the level of the category *car*.

The features of an object to be classified typically cause multiple remindings. PROTOS2 combines these remindings to focus on a region of the category structure. The search for a model begins with the strongest reminding of an exemplar or category and spreads to neighbors in the region until knowledge-based pattern matching succeeds. When reminded of an exemplar, PROTOS2 attempts to establish a match using the exemplar as a model. When reminded of a category, PROTOS2 selects a prototypical exemplar of the category as a model. This selection is based on knowledge of the *family resemblance* and *mass* of each exemplar. The family resemblance of an exemplar is a measure of typicality[ROSC75]. An exemplar is "typical" of a category if it contains features shared by many other exemplars of the category. The mass of an exemplar is the frequency with which it has been seen. An exemplar with high mass represents a collection of objects which are equivalent with respect to the training received by the system. In the absence of idiosyncratic features, PROTOS2 uses the heuristic that an exemplar of high family resemblance and mass will be a useful category model.

After the initial selection of an exemplar based on reminding, PROTOS2 uses three techniques to establish a close match. First, knowledge-based pattern matching attempts to coerce object features to conform to the exemplar, as described in section 3.2. Second, unexplainable differences between the object and the exemplar may index neighboring exemplars which match the object more closely. Difference links between exemplars record the distinguishing features of each exemplar with respect its closest neighbor [McCa81, Kolo83]. These links are learned when new exemplars are added to a category. Third, PROTOS2 focusses interaction with the teacher on the remaining unmatched features. If these techniques result in a strong match then PROTOS2 presents the results to the user.

Sometimes an object to be classified causes multiple remindings. In this event, PROTOS2 considers reorganizing the category structure in three cases:

1) remindings of multiple exemplars within a category. PROTOS2 considers generalizing them to form a single exemplar of higher mass. If domain knowledge and teacher assistance do not enable knowledge-based pattern matching of the exemplars then PROTOS2 seeks additional difference links between them.
2) remindings of multiple categories. PROTOS2 considers generalizing them to form a superordinate category. If common functional features are associated with category membership then PROTOS2 queries the teacher to ascertain their generalization.
3) remindings of multiple exemplars across categories. PROTOS2 considers forming an overlapping category containing the exemplars if the teacher can explain their equivalence.

4. Current Work

Our first concern is the construction and validation of PROTOS2. Building on our implementation of PROTOS1, we intend to construct PROTOS2 in nine months. PROTOS2 will not be a fragile, prototype system. It will be verified by domain experts teaching their heuristic classification knowledge with focussed discussion of examples. The first domain that PROTOS2 will learn is clinical audiology with the assistance of Professor Craig Wier of the University of Texas, Speech and Hearing Department. We will compare the classification performance of PROTOS2 with a hand-crafted, rule-based system that we have built using EMYCIN [Bare84].

Our second concern is research on learning event sequences. Event sequences direct the acquisition of data describing objects. For example, a clinician follows a diagnostic script to gather patient data. This control information is not currently learned by PROTOS, which assumes a fairly complete object description before attempting classification. Preliminary research suggests that knowledge of event sequences can be neatly interleaved with the PROTOS category structure representation of object information. Events are categorized by the (sub)goals they achieve and objects are categorized by their function in enabling events. Psychological research suggests that this larger context for object learning provides cohesiveness to object categories and is learned first [NELS74]. In particular, this context provides a *functional core* for object categories. This will allow PROTOS to build a richer category structure with less guidance from a teacher.

5. Summary

The major contribution of this research is a theory of the acquisition and application of domain knowledge for heuristic classification. The goal of building PROTOS, an inquisitive learner which evolves into an expert, forces a thorough analysis of three fundamental issues in concept formation. First, how are ill-defined, "fuzzy" concepts learned and represented? We believe that induction is not the primary learning mechanism since classical and probabilistic concept representations are inappropriate for most concepts. PROTOS adopts an exemplar-based representation to support the inherent ambiguity of natural concepts. Second, what functions must learned knowledge support? We believe that traditionally the task of object classification has been supported to the exclusion of other tasks. The range of functions that should be supported by concept formation includes summarization of training instances, generation of examples of a concept, prediction of unseen features of a new object, interpretation of "fuzzy" examples, and explanation. PROTOS supports these important applications of learned knowledge by de-emphasizing the role of generalization in concept formation. Third, what are the roles of the teacher and problem solver in concept formation? We believe that learning from classified examples alone is an artificially difficult task. The role of the problem solver is to guide the learner. The role of the teacher is to *explain* problem solving failures. PROTOS is a learning apprentice system which acquires the requisite knowledge for explanation-based, model-driven reasoning.

We are addressing the problems which result from relegating induction to a minor role in learning. Induction compresses data. Without this compression a problem solver can be overwhelmed. The primary problem solving activity in PROTOS is identification of objects by constructing explanations of their similarity to learned concepts. PROTOS focuses this effort on promising candidate matches suggested by the *reminding power* of the new object's features. PROTOS supports the recognition of relevent remindings by learning the importance of features with respect to categories.

We are currently building PROTOS and intend to test its viability for the task of learning heuristic classification knowledge. PROTOS will be instructed by a domain expert and will evolve into an expert system for independent classification and explanation.

REFERENCES

[BARE84] Bareiss, E.R, Bhatttacharjee, A., Rentmeesters, S.J. Help Understand Hearing, an Expert System in the Audiology Domain. Unpublished report of EMYCIN application, University of Texas at Austin, Computer Sciences Department.

[BARE86] Bareiss, E.R. and Porter, B.W. A Survey of Psychological Models of Concept Representation. Forthcoming Technical Report, University of Texas at Austin, Computer Sciences Department.

[BENN85] Bennett, J.S. ROGET: A Knowledge-Based System for Acquiring the Conceptual Structure of a Diagnostic Expert System. *Automated Reasoning 1:1* (1985), 49–74.

[BOOS84] Boose, J. Personal Construct Theory and the Transfer of Expertise. *Proceedings of the National Conference on Artificial Intelligence* (1984), 27–33.

[CLAN83] Clancey, W.J. The Epistemology of a Rule-Based Expert System: A Framework for Explanation. *Journal of Artificial Intelligence*, 20(3) (1983), 215–251.

[CLAN85] Clancey, W.J. Heuristic Classification. *Artificial Intelligence 27*, 289–350.

[COHE85] Cohen, P.R. *Heuristic Reasoning About Uncertainty: An Artificial Intelligence Approach*, (Based on PhD Dissertation, Computer Science Department, Stanford University), Boston: Pitman, 1985.

[DAVI77] Davis, R. Interactive Transfer of Expertise: Acquisition of New Inference Rules. *Proceedings of the International Joint Conference on Artificial Intelligence* (1977), 321–328.

[DEJO81] DeJong, G. Generalization Based on Explanations. *Proceedings of the International Joint Conference on Artificial Intelligence* (1981), 67–69.

[ELLM85] Ellman, T. Generalizing Logic Circuit Designs by Analyzing Proofs of Correctness. *Proceedings of the International Joint Conference on Artificial Intelligence* (1985), 643–646.

[FLAN86] Flann, N.S. and Dietterich, T.G. Selecting Appropriate Representations for Learning from Examples. *Proceedings of the National Conference on Artificial Intelligence* (1986) (to appear).

[HILG75] Hilgard, E.R. and Bower, G.H. *Theories of Learning*, Englewood Cliffs, NJ: Prentice-Hall, Vol. 4, 1975.

[KAHN85] Kahn, G., Nowlan, S. and McDermott, J. MORE: An Intelligent Knowledge Acquisition Tool. *Proceedings of the International Joint Conference on Artificial Intelligence* (1985), 581–584.

[KOLO83] Kolodner, J. L. Maintaining Organization in a Dynamic Long-Term Memory. *Cognitive Science* 7, 4 (1983), 243–280.

[LANG84] Langley, P. and Ohlsson, S. Automated Cognitive Modeling. *Proceedings of the National Conference on Artificial Intelligence* (1984), 193–197.

[LENA77] Lenat, D.B. The Ubiquity of Discovery. *Artificial Intelligence 9:3* (1977), 257-285.

[MAHA85] Mahadevan, S. Verification Based Learning: A Generalization Strategy for Inferring Problem Reduction Methods. *Proceedings of the International Joint Conference on Artificial Intelligence* (1985), 616–623.

[McCA81] McCarty, L.T. and Sridharan, N.S. The Representation of an Evolving System of Legal Concepts. *Proceedings of the International Joint Conference on Artificial Intelligence* (1981), 246–253.

[MICH80] Michalski, R.S. and Chilausky, R.L. Learning by being Told and Learning from Examples: an Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis. *Policy Analysis and Information Systems*, Vol. 4, No. 2 (June, 1980), 125–160. (Special issue on knowledge acquisition and induction).

[MICH83] Michalski, R.S. A Theory and Methodology of Inductive Learning. Appearing in *Machine Learning*, Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (eds.), Tioga Publishing, 1983.

[MINT84] Minton, S. Constraint-Based Generalization: Learning Game-Playing Plans from Single Examples. *Proceedings of the National Conference on Artificial Intelligence* (1984), 251–254.

[MITC82] Mitchell, T.M. Generalization as Search. *Artificial Intelligence*, Volume 18 (1982), 203–226.

[MITC83] Mitchell, T.M., Utgoff, P.E., Nudel, B. and Banerji, R. Learning by Experimentation: Acquiring and Refining Problem Solving Heuristics. Appearing in *Machine Learning*, Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (eds.), Tioga Publishing, 1983.

[MITC85] Mitchell, T.M., Mahadevan, S., and Steinberg, L.I. LEAP: A Learning Apprentice for VLSI Design. *Proceedings of the International Joint Conference on Artificial Intelligence* (1985), 573–580.

[MITC86] Mitchell, T.M., Keller, R.M., and Kedar-Cabelli, S.T. Explanation-Based Generalization: A Unifying View. *Machine Learning 1, 1* (1986).

[NELS74] Nelson, K. Concept, Word and Sentence: Literrelations in Acquisition and Development. *Psychological Review 81* (1974), 267–285.

[POPL82] Pople, H.E.,Jr. Heuristic Methods for Imposing Structure on Ill-structured Problems: The Structuring of Medical Diagnostics. Appearing in *Artificial Intelligence in Medicine*, Szolovits, P. (ed.), Boulder,CO: Westview Press, 1982, pp. 119–190.

[PORT85] Porter, B., Bareiss, R. and Farquhar, A. Learning Domain Knowledge from Fragments of Advice. Appearing in *Recent Progress in Machine Learning*, Mitchell, T. (Ed.), Kluwer Publ., 1985.

[PORT86] Porter, B. and Kibler, D. Experimental Goal Regression: A Technique for Learning Problem Solving Heuristics. *Machine Learning 2* (1986).

[QUIN86] Quinlan, J.R. Induction of Decision Trees. *Machine Learning 1,1* (1986).

[ROSC75] Rosch, E. and Mervis, C.B. Family Resemblance Studies in the Internal Structure of Categories. *Cognitive Psychology 7* (1975), 573–605.

[ROSC78] Rosch, E. Principles of Categorization. Appearing in *Cognition and Categorization*, Rosch, E. and Lloyd, B.B. (ed.), Hillsdale, N.J.:Erlbaum, 1978.

[SCHA82] Schank, R.C. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*, Cambridge University Press, 1982.

[SCHA86] Schank, R.C., Collins, G.C., and Hunter, L.E. Transcending Inductive Category Formation in Learning. *The Behavioral and Brain Sciences* (1986) (to appear).

[SHOR76] Shortliffe, E.H. *MYCIN: Computer-based Medical Consultations*, New York: Elsevier, 1976. (Based on PhD Dissertation, Computer Science Department, Stanford University, 1974.)

[SMIT81] Smith, E. and Medin, D. *Categories and Concepts*, Cambridge: Harvard University Press, 1981.

[STEP86] Stepp, R.E. and Michalski, R.S. Conceptual Clustering: Inventing Goal-Oriented Classifications of Structured Objects. Appearing in *Machine Learning*, Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (ed.), Morgan Kaufman, Vol. 2, 1986, pp. 471–498.

[VALI84] Valiant, L.G. A Theory of the Learnable. *Communications of the ACM 27*, 11 (1984), 1134–1142.

[WINS75] Winston, P.H. Learning Structural Descriptions from Examples. Appearing in *The Psychology of Computer Vision*, Winston, P.H. (ed.), McGraw-Hill, 1975, pp. 157–209. (Based on PhD Dissertation, Computer Science Department, Massachusetts Institute of Technology, Cambridge, MA, 1970.)

[WITT53] Wittgenstein, L. *Philosophical Investigations*, New York: MacMillan, 1953.

Learning Expert Knowledge

by improving the explanations provided by the system

Yves Kodratoff
UA 410 du CNRS, Laboratoire de Recherche en Informatique
Bâtiment 490, Université Paris-Sud
F - 91405 ORSAY

SUMMARY

The word "Learning" is presently used in such a wide variety of meanings that it attracts people that have different or even opposite interests.

This paper starts by a description of what **Expert Knowledge** is, as opposed to **casual knowledge**.

Our main point is that Expert Knowledge is almost as much devoted to efficiency as to explication. Here, we use **explication** in order to differenciate it from the explanations needed in Explanation Based Learning (EBL).
In EBL, the system uses explanations in order to increase its knowledge. The explications we want to obtain are those delivered by the system itself in order to explain its behaviour to its user.
One can guess that EBL is very well suited to improve its explications, but we show that Similarity Based Learning (SBL) is also well able to deliver explications to its users.

Such a system is certainly an Apprentice, in the sense T. Mitchell gives to the word, but it must be even more, we shall call it an **Explicatory Apprentice**.
We show that an Explicatory Apprentice relies on the theorem proving abilities shown by the learning system.

1 - INTRODUCTION

There seems to be a double misunderstanding on the expression "Machine Learning" (called ML in the sequel) that arises between, on the one hand, specialists in Artificial Intelligence (called AI-learnists in the sequel) and, on the other hand, more psychology oriented people (called psy-learnists in the sequel), and non AI oriented learnists (called NOT-AI-learnists in the sequel).

We shall attempt to define what are AI-learnists by qualifying them by two features.

One feature concerns the topic they are working on.

Most of the AI-learnists are working on problems that seem relatively simple because

(first criterion)

there exists a well-known body of knowledge in the field in which learning takes place.

An expert in this field is able to explain to an other expert the reasons of his choices. There may be some disagreement among experts, but the disagreement is on the reasons rather than on the choices themselves.

Typically, Mathematics are expert knowledge, and bicycle riding is casual knowledge.

In every day life, this defines expert knowledge, as opposed to casual knowledge. In the context of ML, we shall add one more criterion to this definition.

The second feature concerns the way they are working on their topic.

There have been recently a considerable emphasis given to the difference between Explanation Based learning (EBL), and Similarity Based Learning (SBL). We disagree on the **emphasis** (not on the difference itself, which is quite real) put on this difference. Our reason is that both EBL and SBL are linked together by a deeply similar approach to Machine Learning. In this paper, we shall see that EBL is rather concerned by the explanation of the successes of a learning system, while SBL is more interested in the explanation of the failures of such a system. But, contrary to what is often thought, we believe that both approaches look for some kind of explanations, using different techniques.

EBL usually uses goal regression techniques, while SBL usually looks for recognition functions obtained by a generalization from a set of examples and counter-examples.

(second criterion)

Both EBL and SBL are characterized by the fact that their results are **expressed in the language of the expert him/herself**.

As opposed to EBL and SBL that belong to AI-learning, there exists also an other approach to ML, illustrated by statistics, and more recently by the connectionist approach [Touretzky & Hinton 1985], where the aim is **efficiency only**, and where no explanation can be expressed in the vocabulary of the expert.

Statistics do provide explanations to their results, but, typically, these reasons are expressed in terms of "quadratic squares", or other statistical concepts. They are expressed in the vocabulary of the expert in statistics, not in the vocabulary of the expert of the field on which statistics are done.

All the EMYCIN like expert systems, that are well-known for providing bad explanations of their behavior because the way they combine positive and negative beliefs, nevertheless provide these explanations in the vocabulary of the expert in the field under study.

Definition.

We shall say that one is doing **expert knowledge acquisition** when the two follow-

ing requirements are fulfilled.
Firstly, the acquisition concerns a field which possesses a body of theory such that the expert in this theory can provide explanation.
Secondly, the acquisition provides explanations of its behavior and uses the vocabulary of the human experts.

When one of this condition is not fulfilled, then we define the acquisition of knowledge as **casual knowledge acquisition**.

Let us give three examples of NOT-AI-learning, ordered by their distance to AI-learning.

Riding a bicycle is an example of every day life casual knowledge, its learning therefore belongs to casual knowledge acquisition.

A diagnosis system that would rely on thousends of clinical cases, store all of them, and provide a diagnosis by a template matching mechanism does not provide explanations, therefore it belongs to NOT-AI-learning. It is similar to rote learning which clearly does not concern AI specialists.

A diagnosis system that uses pure numerical techniques in order to perform its clusterings, and generate its recognition functions, cannot give explanations in the vocabulary of the expert.

There are here some (fortunate !) shadows on the limits between AI and NOT-AI-learning, since the clustering algorithm may or may not include, as parameters, some semantics of the field.
Michalski's "conceptual clustering" [Michalski & Stepp 1983] is very typical of a numerical technique that falls into AI-learning because it can provide some explanations of its clusters in the expert's vocabulary.

The NOT-AI-learnists are interested in casual knowledge acquisition because they do not mind explanations but efficiency.
The psy-learnists mind explanations but they are interested in the way humans actually **store** their knowledge, which seems to be very far from the way experts **explain** their knowledge. Obtaining explanations from an expert being always a painful process, one can guess that it is stored in some way that can be qualified of casual, in the sense we give here to this word.

One necessary condition to the generation of explanations is that one is (i.e., the system is) able to prove that its actions obey some constraints _the quality of the proof is an other matter that will be seen later.

As an introduction let us first see how proving things may be a first step towards explanations.
We want also to illustrate that, contrary to the "Explanation Versus Similarity Based Learning" way of looking at Machine Learning, Similarity Based Learning must also provide explanations, and needs some theorem proving.

EBL is born from techniques that are efficient on a very well defined domain.
For instance, the recently defined EBG (G for Generalization), [Mitchell & Al. 1986], requires both
- a complete theory,

- a definition of the concept under learning.

It would be useful to be able to define generalization in a domain where the concept to be reached is still unknown, or the theory still to be completed.
This has been done by Michalski [Michalski 1983, 1984], in his formalization of SBL.
Our aim is to push forward this theory, and attempt to encompass both EBL and SBL into it, by showing that SBL also should provide explanations of its generalizations.
We hope to reach two different goals.
The first is to show how a formal theorem prover can help in providing explanations that improve the concept under learning.
The second is to show how much these formalities are actually simple, and in many cases, easy to use, once one disposes of a theorem prover. The most widely available being the language PROLOG, we used its formalism in the sequel.

2 - NOTATIONS

In this paper, we shall use a PROLOG-like notation of the clauses.

2.1 - Representation of "static" knowledge

Let us use "All human beings are mortal etc ..." as example.
These natural language sentences may have several logical representations.
Let us assume that they can be represented by the theorem and the two assertions

$$\forall x\ [\mathrm{HUMAN}(x) \Rightarrow \mathrm{MORTAL}(x)]$$
$$\mathrm{HUMAN}(\mathrm{SOCRATES})$$
$$\mathrm{MORTAL}(\mathrm{SOCRATES})$$

Everyone is used to representation by IF ... THEN ... expressions, where the theorem takes the form

IF HUMAN(x) THEN MORTAL(x)

with an implicit universal quantification.
Instead of writing ' IF HUMAN(x) THEN MORTAL(x) ', one can write ' MORTAL(x) IF HUMAN(x) ', this is what is done in a PROLOG representation.
In this paper, following PROLOG representation, this theorem will be written as

```
MORTAL(x)     :-           HUMAN(x)
          equivalent to
MORTAL(x)     IF           HUMAN(x)
```

where the ' :- ' is nothing but an ' IF '.

More generally, a Horn clause has the form

A :- B, C, D, ...

which means, A is TRUE IF (B is TRUE, and C is TRUE, and D is TRUE, ...).
A is called the **conclusion** of the clause, and B, C, D, ... are called the **conditions** of the clause.
A clause without condition, i.e. a pure positive clause is often called a fact, or a data.

A clause without a conclusion, i.e. a pure negative clause, is called a **question**.

2.2 - Representing inference

It is clear that nothing special is brought by this representation as such. Its interest comes from the fact that an inference mechanism is included in it.
Given the clauses

(C_1) MORTAL(x) :- HUMAN(x)
(C_2) HUMAN(SOCRATES) :-

In general, one should be able to resolve C_2 and C_1, which means that one should remark that C_2 fulfills the condition of C_1, x being intantiated by SOCRATES.
Using PROLOG representation implicitly says that this kind of reasoning, usually called "forward chaining", will not be used. On the contrary, one will only use "backward chaining", i. e. a question (i.e. a pure negative clause) will have to match some conclusions and will generate new questions etc ... up to the moment where all questions have been answered.
PROLOG uses a refutation strategy, therefore a question will be stated in the form of a pure negation, as already defined.
The original question and all sub-questions have been answered when all of them have been put in contradiction with some parts of the data basis. One then says that one has **derived the empty clause** from the data basis and the question, i.e. that the question is inconsistent with the data basis.

In our example, the only way to deduce something about Socrates' mortality, is asking the question "Is Socrates mortal ?" by adding a pure negative clause in the form

(C_3) :- MORTAL(SOCRATES)

Clause C_3 is a pure condition clause that matches the conclusion of C_1, therefore generating the new question

(C_4) :- HUMAN(SOCRATES)

Since C_4 and C_2 contradict each other, one concludes that C_3, which is the pure negation of the possibility for Socrates to be mortal, is contradictory with the other clauses, viz. C_1 and C_2. By the classical refutation proof argument, one concludes that the following set is coherent

(C_1) MORTAL(x) :- HUMAN(x)
(C_2) HUMAN(SOCRATES) :-
(C_2) MORTAL(SOCRATES) :-

This representation implicitly contains a theorem prover (reduced to Horn clauses, with one only pure negative clause). This is why we have chosen this representation : it contains no ambiguity on the way inference will be performed.

3 - IMPROVING A GENERALIZATION

In the following examples, we shall suppose that we reached so far a given state of knowledge acquisition, and that we are trying to improve it, by checking the present state of the learned recognition function against new examples or counter-examples. When they do not fit together, the problem is then to be able to improve this recognition function.

Said in an intuitive way, one expects from a recognition function to "recognize" new examples and to "reject" new counter-examples.
Let us now give one possible definition for recognition and rejection.

We choose here a quite intuitive way of defining these words. More details have been given elsewhere. This definition has also been extensively used by J. Nicolas [Nicolas 1986].

The reader is asked to accept these definitions as temporary hypotheses : other definitions would lead to other proofs, but the essential step that we want to illustrate here, viz. how a proof can be the basis for an explanation, does not depend on these definitions.

Let E, CE, and f(x) respectively be an example, a counter-example, and a recognition function.

3.1 - Recognition of an example

One says that a function **f(x) recognizes an example** E when there is no contradiction to assert that both E and $\exists x\, [f(x)]$ are TRUE.

We say that it is an "intuitive" definition because of the following argument.
Suppose that f(x) is a recognition function in the usual meaning of the word, i.e. E is an instance of f(x), i.e. there is an instance of ' x ', say $\{x \leftarrow A\}$ such that $f(A) = E$.
Then, if there is a contradiction between E and $\exists x\, [f(x)]$, there is no contradiction between E and $\neg\exists x\, [f(x)]$.
But this is to say that E and $\forall x\, [\neg f(x)]$ is coherent, i.e. that f(A) and $\forall x\, [\neg f(x)]$ is coherent, which is a contradiction.

Example 1.
Let us suppose that we have so far obtained the following recognition function

$$f_1(x, y) = SPHERE(x)\ \&\ RED(y)$$

which means : "there are two objects, one is a sphere, the other one is red".
Suppose that a further example is given

$$E_1 = SPHERE(A)\ \&\ RED(A)$$

where ' A ' is the name of an object which happens to be a red sphere,
then E_1 is clearly an instance of $f_1(x, y)$ since the substitution

$$\sigma_1 = \{x \leftarrow A,\ y \leftarrow A\}$$

is such that

$$\sigma_1 f_1(x, y) = E_1.$$

One can therefore prove that $E_1 \Rightarrow \exists x\, y\, [f_1(x, y)]$ since the set of clauses

$$\begin{array}{lll} C_1 : & SPHERE(A) & :- \\ C_2 : & RED(A) & :- \\ C_3 : & & :-\ \ SPHERE(x),\ RED(y) \end{array}$$

is contradictory.
SPHERE(x) in C_3 resolves with C_1, and RED(y) in C_3 resolves with C_2. This shows that $\{C_1, C_2, C_3\}$ is contradictory. One usually says that $\{C_1, C_2, C_3\}$ leads to the empty clause with the substitution σ_1 above.

Clause C_3 is equivalent to $\forall x\, y\ \neg[f_1(x, y)]$, which is the negation of $\exists x\, y\ [f_1(x, y)]$.

The above example illustrates why one can use the definition of "recognition" we just gave, but still does not explains its use, since the direct proof by substitution is possible.
It may happen that the substitution is very hard or impossible (as example 2 shows) to find because, in order to make sure that E is an instance of f(x), one must use semantic properties of the micro-world one is learning in.
In that case, it may be easier to prove that one can deduce $\exists x\ [f(x)]$ from E.

Example 2.
Suppose now that the recognition function is

$$f_2(x) = ELLIPSOID(x)\ \&\ RED(x)$$

which means that we have memorized that "there is a red ellipsoid" in all the scenes we are learning from.
Suppose that we are given a further exemple

$$E_2 = SPHERE(B)\ \&\ RED(B)$$

where ' B ' is the name of a red sphere.
Since ELLIPSOID and SPHERE do not match, the proof by substitution is useless.
Suppose now that the semantics of the micro-world where learning is taking place are such the following theorem is known

$$TH_1 : \forall x\ [SPHERE(x) \Rightarrow ELLIPSOID(x)]$$

We shall attempt to prove that $E_2 \Rightarrow \exists x\ [f_2(x)]$, and add TH_1 in our knowledge base. This reads

C_4 :	SPHERE(B)	:-	
C_5 :	RED(B)	:-	
C_6 :	ELLIPSOID(x)	:-	SPHERE(x)
C_7 :		:-	ELLIPSOID(x), RED(x)

C_7 resolves with C_6, leading to the new clause

$$C_8 : \quad :- \quad SPHERE(x), RED(x)$$

and C_8 resolves with C_4 and C_5 to lead to the empty clause with the substitution $\{x \leftarrow B\}$.
Therefore, E_2 is recognized by $f_2(x)$.

3.2 - Rejection of an example

One says that a function **f(x) rejects an example E** when, asserting that both E and $\exists x\ [f(x)]$ are TRUE, is a contradiction.

Intuitively, it is clear one want to define the case where a recognition function rejects a counter-example CE. It follows that CE and $\exists x\ [f(x)]$ should be contradictory. Therefore, CE and the negation of $\exists x\ [f(x)]$ should not be contradictory.

and CE and $\forall x\,[\neg f(x)]$ should not lead to a contradiction.

Example 3.
Suppose that the recognition function is

$$f_3(x) = SPHERE(x)\ \&\ RED(x)$$

and that

$$CE_3 = SPHERE(C)\ \&\ RED(D)$$

is a counter-example to f_3.
Clearly, CE_3 *is not an instance of* f_3 *since* x *cannot be substituted by both* C *and* D.
Using the above formalism, one checks that the system of clauses

$$\begin{array}{llll} C_9: & SPHERE(C) & :- & \\ C_{10}: & RED(D) & :- & \\ C_{11}: & & :- & SPHERE(x),\ RED(x) \end{array}$$

does not lead to the empty clause since, if one resolves C_{11} *with* C_9, *one obtains*

$$\begin{array}{llll} C_{10}: & RED(D) & :- & \\ C_{12}: & & :- & RED(C) \end{array}$$

because ' x ' has been instantiated by ' C ' during the resolution. C_{10} *and* C_{12} *cannot be reduced since ' C ' and ' D ' are different constants.*
Since C_{11} *is the clause form of* $\forall x\,[\neg f_3(x)]$, *this proves that* CE_3 *is a counter-example for* $f_3(x)$.

All the above illustrates our definitions and the proof procedure but is not very significant as explanations generator. The reason is that the definitions work well in these cases, therefore the expected success or failure of the proof that E (respectively CE) allows to deduce (resp. not to deduce) the theorem $\exists x\,[f(x)]$, just "explains" why E (resp. CE) is an example (resp. a counter-example).

3.3 - Improvment of a formula by an explanation of its success to recognize a new example.

The problem is to explain why the proof succeeds.
This process can be interesting in two cases.
First case. It may be that the recognition function is too much "hairy", i.e., it contains irrelevant information that does not harm the recognition of the given example, but could be harmful in other situations.

For instance, imagine a recognition function for ' man ' that contains a predicate ' beard ' taking the value TRUE when the man has a beard.
It may be that only bearded men have been seen so far by the system, which recognizes ' man ' only if its description gives the value TRUE to ' beard '.
- Given a complete theory of ' man ', in this case, a detailed description of what are secondary sexual features,
- Given a complete description of a bearded man,
the system should be able to prove that the predicate ' beard ' is not necessary to assert that this bearded man is actually a man, because it is a secondary feature only.

As this example shows, since irrelevant information is dropped, another consequence is that some generalization is performed on the given formula.
This is the topic of "goal regression" [Waldinger 1977, Nilsson 1980], and of

Explanation Based Generalization [Mitchell & Al. 1986].

We shall use here a PROLOG version of the "safe-to-stack" example, taken from [Mitchell & Al. 1986].

The PROLOG program that contains the information relative to the fact that BOX_1 can be safely stacked on $ENDTABLE_1$ is contained in the following set of clauses.
Most clauses are a direct rewritting of those in [Mitchell & Al. 1986].
Three differences must be pointed at.
The first difference is that we use integers, therefore we multiply by ten the values given in Mitchell et Al. paper.
The second difference is that the "ISA" links are dropped. This would have to be expressed as types in a typed PROLOG, and typing PROLOG is not our present topic.
The third difference is that the default value is given in clause C_{12} as a "normal" value. The fact that it is a default value is expressed by clause C_{11}, just before C_{12}. Clause C_{11} computes the weigths and, only if it fails to compute, will allow to give its default value to the weight of an endtable.

C_1	ON(BOX_1, $ENDTABLE_1$)	:-	
C_2	COLOR(BOX_1, RED)	:-	
C_3	COLOR($ENDTABLE_1$, BLUE)	:-	
C_4	VOLUME(BOX_1, 10)	:-	
C_5	DENSITY(BOX_1, 1)	:-	
C_6	FRAGILE($ENDTABLE_1$)	:-	
C_7	OWNER($ENDTABLE_1$, CLYDE)	:-	
C_8	OWNER(BOX_1, BONNIE)	:-	
C_9	SAFE-TO-STACK(x, y)	:-	NOT FRAGILE(y)
C_{10}	SAFE-TO-STACK(x, y)	:-	LIGHTER(x, y)
C_{11}	WEIGHT(x, w)	:-	VOLUME(x, v), DENSITY(x, d), w is v*d,
C_{12}	WEIGHT($ENDTABLE_1$, 50)	:-	
C_{13}	LIGHTER(x, y)	:-	WEIGHT(x, w_1), WEIGHT(y, w_2), LESS(w_1, w_2)
C_{14}	LESS(x, y)	:-	$x < y$
C_{15}		:-	SAFE-TO-STACK(BOX_1, $ENDTABLE_1$)

The pure negative clause C_{15} asks the question wether it is safe to stack a given BOX_1 on a given $ENDTABLE_1$.

The proof proceeds as the following trace shows. This trace is provided by most PROLOG interpreters.
The comment ' Call ' means that the predicate have been used as a question to the system.
The comment ' Exit ' means that the the predicate (with the instances in the ' Exit ') has been proven TRUE.
The comment ' Fail ' means that the predicate has been proven FALSE.
The comment ' Back to ' means that back-tracking is taking place.
The numbers to the left are those provided by the compiler, we have put on the right the level of embedding they actually represent.

For instance, Call ' NOT FRAGILE($ENDATBLE_1$) ' is indexed by ' 2 ' and call LIGHTER(BOX_1, $ENDTABLE_1$) is indexed by ' 4 ' in the computer output. Actually, they are at the same level of embedding and are labelled respectively 2_1 and 2_2 on the right.

```
1   Call:      SAFE-TO-STACK(BOX1, ENDTABLE1) ------------------------+
2   Call:      NOT FRAGILE(ENDTABLE1) ---------------------+          |
3   Call:      FRAGILE(ENDTABLE1)                          |          |
3   Exit:      FRAGILE(ENDTABLE1)                          21         |
2   Back to:   NOT FRAGILE(ENDTABLE1)                      |          |
2   Fail:      NOT FRAGILE(ENDTABLE1) ---------------------+          |
4   Call:      LIGHTER(BOX1, ENDTABLE1) ---------------------+        |
5   Call:      WEIGHT(BOX1, w1) ------------------+          |        |
6   Call:      VOLUME(BOX1, v1) ----------+       |          |        |
6   Exit:      VOLUME(BOX1, 10) ----------+ 41    |          |        |
6   Call:      DENSITY(BOX1, d1) ---------+       31         |        |
6   Exit:      DENSITY(BOX1, 1) ----------+ 42    |          |        1
6   Call:      w1 is 10*1 ----------------+       |          |        |
6   Exit:      10 is 10*1 ----------------+ 43    |          |        |
5   Exit:      WEIGHT(BOX1, 10) ------------------+          |        |
5   Call:      WEIGHT(ENDTABLE1, w2) -------------+          22       |
6   Call:      VOLUME(ENDTABLE1, v2)              |          |        |
6   Fail:      VOLUME(ENDTABLE1, v2)              32         |        |
5   Back to:   WEIGHT(ENDTABLE1, w2)              |          |        |
5   Exit:      WEIGHT(ENDTABLE1, 50) -------------+          |        |
5   Call:      less(10, 50) ----------------------+          |        |
6   Call:      10<50 ---------------------+       33         |        |
6   Exit:      10<50 ---------------------+ 44    |          |        |
5   Exit:      less(10, 50) ----------------------+          |        |
4   Exit:      lighter(BOX1, ENDTABLE1) ---------------------+        |
1   Exit:      SAFE-TO-STACK(BOX1, ENDTABLE1) ------------------------+
```

Let us now analyse the above proof and show that it actually gives a set of explanations why it is safe to stack BOX_1 on $ENDTABLE_1$, which, in the rest of this section will be abreviated by "explanation".

A level always begins with a question, labelled as a ' Call '. When it succeeds, it ends with an ' Exit '. The exit contains the reason why the call succeeded. This why one can say that each level provides an "explanation", that becomes more an more refined as one goes down the levels.

Level 1 is the most outside. In a sense it says
"it is safe to stack BOX_1 on $ENDTABLE_1$" because I have proven it just now. It is the most superficial level of explanations, the children use quite often !

level 2 contains sub-level 2_1 and sub-level 2_2.
Sub-level 2_1 is a failure sub-level : it tells that "NOT FRAGILE" has nothing to do with "the explanation". We disregard it now, but one must be aware that, when explanations for negative features are looked for, then their explanation is given by the failure sub-levels only.
As an exercice, the reader can analyse the trace obtained by giving the default value ' 10 ' to the weight of ENDTABLE$_1$, and ask the question

:- NOT SAFE-TO-STACK(BOX$_1$, ENDTABLE$_1$)

The answer is then also ' yes ' but the analysis of the explanations why it is not

safe to stack BOX_1 on $ENDTABLE_1$ is completely different.
Sub-level 2_2 provides the "explanation" :
"it is safe to stack BOX_1 on $ENDTABLE_1$" because BOX_1 is lighter than $ENDTABLE_1$".

Level 3 contains three sub-levels 3_1, 3_2, and 3_3. The explanations obtained from each one must be conjuncted to obtain the "explanation".
They provide the "explanation" :
"it is safe to stack BOX_1 on $ENDTABLE_1$" because
the weight of BOX_1 is 10, the weight of $ENDTABLE_1$ is 50, and 10 is less than 50.

One can be tempted to generalize at once by saying that the weight of BOX_1 is w_1, the weight of $ENDTABLE_1$ is w_2, and w_1 is less than w_2. This is not allowed by EBG that says that one can generalize further only if the numerical values come from example data. If some numerical valuethere is issued from theory data, then this value should be kept as such.
In this case, the default value : the weight of $ENDTABLE_1$ = 50, is part of the theory data, not of the example data.
An other way to look at this is to say that one must keep them when there is no deeper explanation to the numerical values. In this case, there is no deeper explanation to the fact that the weight of $ENDTABLE_1$ = 50, since sub-level 3_2 contains no inner sub-level. Therefore, this value will be kept in the final result.

Sub-level 3_1 says that the weight of BOX_1 is 10 because its volume is 10, its density is 1, and because 10*1 = 10. In this case, the numerical values are issued from the example and can be generalized.
The value of the volume is called v_1, the value of the density is called d_1, which gives the "explanation" :
The weight of BOX_1 is $w_1 = v_1 * d_1$.

Sub-level 3_3 contains an explanation given by 4_4. This explanation is disregarded because it uses a function, like <, of low level. Deciding what is at what is not "low level" function is a quite easy decision that must always be taken beforehand.

Applying this generalization into the explanation of level 3 (which is the last "explanation" found) leads to the final "explanation" :
"it is safe to stack BOX_1 on $ENDTABLE_1$" because
the weight of BOX_1 is $w_1 = v_1 * d_1$, the weight of $ENDTABLE_1$" is 50, and w_1 is less than 50.

The process we describe here, is nothing but a paraphrasing of EBG, with two differences with the original paper.
Our presentation has a stronger theorem proving orientation.
Instead of forcing the variables down to elementary facts, we force the constants up to some level where they can be generalized. In an implementation, EBG is the correct way to realize the transmission of the relations among variables. We felt neverthless easier to understand why this process is an explanation of something when presented the other way round.

DeJong and oney recently presented a discussion of EBG. We shall not comment here on their criticism [DeJong and Mooney 1986] except on the one concerning the case where two or more explanations are possible. This point will be detailed in section 4, since it arises also in the context of explaining the failures.

At any rate, as useful as it is, on must stress that goal regression hardly provides the possiblity for a progressive improvment of the quality of the explanations. This will be possible only when the theory itself will be improved : explanations for failures are necessary to improve the theory.

3.4 - Improvment of a recognition function by an explanation of its failure to recognize a new example.

Let us suppose that f(x) cannot recognize a new example E. This means that one cannot deduce $\exists x\, [f(x)]$ from E, therefore the proof that E and $\forall x\, [\neg f(x)]$ fails, i.e. one cannot deduce the empty clause.
The problem is now to **explain why the proof fails**. This process is usually difficult to implement, as the following example shows.

Example 4
Let

$$f_4(x) = SPHERE(x)\ \&\ RED(x)$$

and suppose that a new example is

$$E_4 : SPHERE(E)\ \&\ RED(F)$$

One fails to prove that one can deduce $\exists x\, [f_4(x)]$ from E_4, as the following set of equivalent clauses shows.

C_{13} :	SPHERE(E)	:-	
C_{14} :	RED(F)	:-	
C_{15} :		:-	SPHERE(x), RED(x)

In example 3, $C\dot{E}_3$ was considered as a counter-example to f_3, therefore the failure of the proof was just normal. Now we must find an explanation to the failure. This is more or less equivalent to find a new function, $f_4'(x)$, that is "the closest possible" to f_4, but allows the proof to succeed.

The failure can be issued from two very different reasons. Either there is a problem with the predicates themselves (one cannot find a predicate in the conclusion of the clauses (further called conclusion-predicate) to match an other one in the conditions of the clauses (further called condition-predicate)), or there is a problem with the substitution.

Example 4 illustrates the second case. Both condition-predicates SPHERE and RED can match their conclusion counter-parts in C_{13} and C_{14}, but 'x' must be instantiated either by 'E' or by 'F'.

Imagine that, instead of C_{15}, the above set of clause would contain instead

$$C_{15}' : \quad :-\ SPHERE(x),\ RED(y),\ RELIGIOUS(z)$$

Then the variables 'x' and 'y' could be correctly instantiated by E and F, but no conclusion-predicate could match 'RELIGIOUS'. This would then become the reason of the failure of the proof.

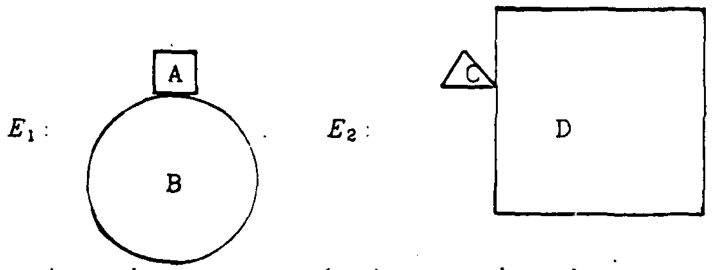
When the reason of the failure of the proof is a substitution problem, then one has to introduce variables at the right places to insure the success of the proof with the new generalization.
When the reason of the failure of the proof is a predicate problem, it can be

easily found in some cases where one only misses, in the middle of many others that match. Nevertheless, in general, it is very difficult to explain the failure.
When the reason of the failure of the proof mixes substitution and predicate problems, then finding the reason of the failure becomes more or less untractable.

This why we have developed an algorithm, called AGAPE, and described elsewhere [Kodratoff 1983, Kodratoff & Al. 1985, Kodratoff & Ganascia 1986], the role of which is to trace down the possible failures in a given set of examples. The central mechanism for this has been called Structural Matching : it preserves as much as possible the structure of the examples before attempting any generalization.

3.5 - A simple example of Structural Matching (SM).

Consider the two following examples.



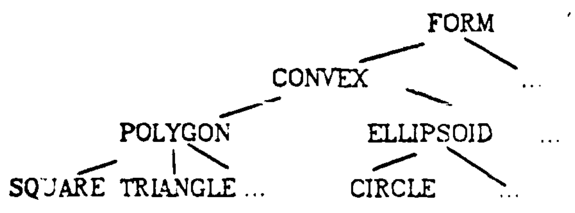
Using his intuition, the reader may notice that he can find two different generalizations from these examples.
He sees that either
- there are two different objects touching each other, and a small polygon
- there are two different objects touching each other, one of them is a square.

Both generalizations are true and there is no reason why one of them should be chosen rather than the other. We shall now see that one of the interesting features of SM is that it keeps all the available information, and therefore constructs a formula containing both the above two "concepts".

The examples can be described by the following formulas

$$E_1 = \text{SQUARE(A) \& CERCLE(B) \& ON(A, B) \& SMALL(A) \& BIG(B)}$$
$$E_2 = \text{TRIANGLE(C) \& SQUARE(D) \& TOUCH(C, D) \& SMALL(C) \& BIG(D)}$$

Let us suppose that the following hierarchy is provided to the system.



together with the theorems

$$\forall x \forall y \, [\mathrm{ON}(x, y) \Rightarrow \mathrm{TOUCH}(x, y)]$$
$$\forall x \forall y \, [\mathrm{TOUCH}(x, y) \Leftrightarrow \mathrm{TOUCH}(y, x)]$$

This taxonomy and the theorems represent our semantical knowledge about the micro-world in which learning is taking place.

The SM of E_1 and E_2 proceeds by transforming them into equivalent formulas E_1' and E_2', such that E_1' is equivalent to E_1, and E_2' is equivalent to E_2 in this micro-world (i.e., taking into account its semantics).
When the process is completed, E_1' and E_2' are made of two parts.
One is a variabilized version of E_1 and E_2. It is called the **body** of the SMized formulas. When SM succeeds, the bodies of E_1' and E_2' are identical.
The other part, called the **bindings** (of the variables), gives all the conditions necessary for the bodies to be identical.

In our example,

Body of E_1' =
POLYGON(u, y) & SQUARE(x) & CONVEX(ν_1, ν_2, z) & ON(y, z) & TOUCH(y, z) & SMALL(y) & BIG(z)
Bindings of E_1' =
((x = y) & (y ≠ z) & (x ≠ z) & (ν_1 = ELLIPSOID) & (ν_2 = CIRCLE) & (u = SQUARE) & (x = A) & (z = B))

Body of E_2' =
POLYGON(u, y) & SQUARE(x) & CONVEX(ν_1, ν_2, z) & TOUCH(y, z) & SMALL(y) & BIG(z)
Bindings of E_2' =
((x ≠ y) & (y ≠ z) & (x = z) & (ν_1 = POLYGON) & (ν_2 = SQUARE) & (u = TRIANGLE) & (x = D) & (y = C))

The algorithm that constructs E_1' and E_2' is explained in [Kodratoff 1983, Kodratoff & Ganascia 1986, Kodratoff & Al. 1985].
The reader can check that E_1' and E_2' are equivalent to E_1 and E_2.

E_1' and E_2' contain exactly the information extracted from the hierarchy and the theorems which is necessary to put the examples into SM.

For instance, in E_1', the expression ' (POLYGON(u, y) ' means that there is a polygon in E_1, and since we have the binding (u = SQUARE), it says that this polygon is a square, which is redundant in view of the fact that SQUARE(x) & (x= y) says that x is a square and is the same as y. This redundancy is not artificial when one considers the polygon in E_2 which is a TRIANGLE.

Once this SM step has been performed, the generalization step becomes trivial : we keep in the generalization all the bindings common to the Smized formulas and drop all those not in common.

The generalization E_1 and E_2 is therefore

E_g : POLYGON(u, y) & SQUARE(x) & CONVEX(ν_1, ν_2, z) & TOUCH(y, z) & SMALL(y) & BIG(z)
with bindings (y ≠ z).

In "English", this formula means that there are two different objects (named y and z), y and z touch each other, y is a small polygon, z is a big convex, and there is a square (named x) which may be identical to y or z.

AGAPE works on a given set of examples and is not tuned to incremental learning. Nevertheless, elementary changes would make it work incrementally, as long as the structural matching is preserved. An explanation of each change due to a new example would then be possible.

In other words, it would not be too difficult to include AGAPE into an apprentice system as long as the "good" structural matching has been found with the first set of examples.
In that case, the explanations provided by AGAPE would be of increasing quality as the number of new examples increases.
On the contrary, if a new example differs widely from the present generalization, then a completely new generalization process would have to take place, thus providing no explanations.

3.6 - Improvment of a recognition function by an explanation of its failure to reject a new counter-example.

Let us suppose that f(x) recognizes a new counter-example CE. This means that the proof that CE and $\forall x$ [-f(x)] succeeds instead of failing as it should if the counter-example would be rejected.
Explaining a success, if not easy, is usually less complicated than explaining a failure. This is why we think that a method can be devised for incremental learning in this case.

Example 5.
Suppose that the recognition function is

$$f_5(x, y) = SPHERE(x) \ \& \ RED(y)$$

and that

$$CE_5 : SPHERE(G) \ \& \ RED(G)$$

is a counter-example to f_5.
Writing it as clauses, one sees at once that one cannot deduce $-f_5$ from CE_5.

C_{16} :	$SPHERE(G)$	:-	
C_{17} :	$RED(G)$	:-	
C_{18} :		:-	$SPHERE(x), RED(y)$

leads to the empty clause with the substitution $\{x <- G, y <- G\}$.

As one can see the success of the proof is easy to explain : any substitution that leads to the result is a kind of "explanation".
In the example, but also in general, the simpler way to change f_5 in order to obtain a new function f_5', that forbids the success of the proof, is to forbid the substitutions that lead to a success. Since a substitution can be represented by an equality, the new function will be obtained from the old one, by adding the condition that its variables do not take the values as in the substitution.

From f_5 and the new counter-example CE_5, one obtains the new recognition function

$$f_5(x, y) = SPHERE(x) \& RED(y) \& -[[x = G] \& [y = G]]$$
which means that one must forbid to ' x ' or to ' y ' to take the value ' G '.

All that can be felt as hugely far from anything that will learn incrementally, and even further from a system that improves its explanations. The following will show how much this feeling is wrong.

3.7 - Relative role of examples and counter-examples.

The examples express the fact that there exists this or that property common to the examples. On the contrary, the counter-examples express the fact that none of the examples possess this or that property. Therefore, the examples must verify the theorem obtained by negating the one which best expresses the properties of the counter-examples.
If a more general theorem is chosen, say T", which does not imply all the other theorems that can be deduced from the counter-examples, then there could exist some examples, say x_0, for which
$$-[f(x_0) => -T''(x_0)]$$
even if the "correct" $f_C(x_0)$ is such that
$$[f(x_0) => -f_C(x_0)].$$

We reach the conclusion that examples allow us to find recognition functions and the counter-examples allow us to find theorems that must be verified by all TRUE instances of the recognition function.

If $\exists x\ [f_C(x)]$ is this best theorem deduced from the counter-examples, then the instances of x that belong to the examples must verify $-\exists\ [f_C(x)]$. That is to say,

one has to prove the following :
$$\forall x\ [f(x) => -f_C(x)]$$
in order to verify that the domain of the examples and the domain of the counter-examples do not overlap.
Note that this formula is valid only for an f(x) which is supposed to be the "best" one, i.e. "the" one that characterizes the domain of the examples
On the contrary, it will be possible to check this formula for any property, $f_C(x)$ of the counter-examples.
An other remark is that one must not be suprised when "$f_C(x)$ does not depend on x. This is always the case when one dispose of one counter-example only, since no variablization can have taken place.

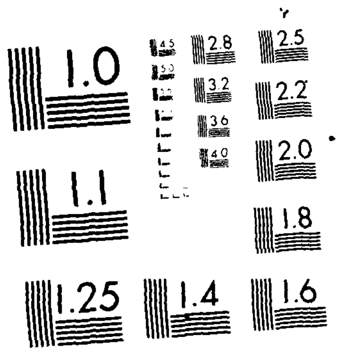
For the remainder of this section, let
$$f_6(x, y) = SPHERE(x) \& BLACK(y).$$
Suppose that the counter-example is
$$CE_6 = [BLACK(A) \& SPHERE(A)],$$
the best theorem is CE_6 itself, the theorem that must be verified by the examples is $-CE_6$, i.e.
$$-[BLACK(A) \& SPHERE(A)].$$
One must attempt to prove
$$\forall x, y\ [\ [BLACK(x) \& SPHERE(y)] => -[BLACK(A) \& SPHERE(A)]\ ]$$ *and one will, of course, fail.*

We have already described what must be done if the proof succeeds, we shall skip it here.

AD-A173 886 PROCEEDINGS OF THE INTERNATIONAL MEETING ON ADVANCES IN LEARNING HELD IN..(U) CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE ORSAY (FRANCE) L.. D J GILMORE ET AL. 3/3

UNCLASSIFIED 01 AUG 86 RR-290 F/G 5/10 NL

END
DATE
FILMED
1-87
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

If the proof fails, this means that the counter-examples intersect the examples, one must attempt to construct a new generalization $f(x)$ which implies $-f_C(x)$. We shall now propose a method for doing so. It uses an attempt to prove a particular theorem, henceforth called Th. The reason why Th is chosen cannot be understood beforehand, the reader is asked to wait a little before he can see the interesting consequences of its proof.

$$\textbf{Th} : \exists x\, [f(x) \Rightarrow f_C(x)].$$

Let us attempt to prove that $\exists x, y\, [\, [BLACK(x)\ \&\ SPHERE(y)] \Rightarrow [BLACK(A)\ \&\ SPHERE(A)]\,]$

There are three cases.

First case : Th is not provable.

The reason may be that $\forall x\, [f(x) \Rightarrow -fC(x)]$ has been proven. In that case, the new property of the counter-examples actually does not cover any example, and nothing has to be changed.

It may also be so we are unable to prove both Th and $\forall x\, [f(x) \Rightarrow -fC(x)]$. This is the failure case, where nothing can be learned. It shows that we ignore an essential property of the domain, but does not tell where to find it.

Second case : Th is provable and reduces to the empty clause.

We prove it by refutation, attempting to prove that one can deduce the empty clause from -Th. Since Th is provable, one will succeed and each success delivers a substitution σ which is an instance of the substitutions to be made to x in order to verify Th. By carrying out all the possible proofs, in the case where there is a finite number of them, or, in infinite cases, by inventing a function that covers all the cases (this part is not emphasized here but is a difficult part of learning), one defines the set of all the x_i such that $f(x_i) \Rightarrow f_C(x_i)$. Let us call P_i this set :

$$P_i = \{\, x_i \,/\, f(x_i) \Rightarrow f_C(x_i)\, \}.$$

We now claim that

$$\forall x\, [\, [f(x)\ \&\ -[x \in P_i]] \Rightarrow -f_C(x)\,].$$

In other words, we have found the $f(x)$ we have been looking for.
Proof.
Since x belong to the set for which the implication $f(x_i) \Rightarrow f_C(x_i)$ is FALSE and in all the cases where $f(x)$ is TRUE, it follows that $f_C(x)$ is FALSE.

In order to prove
$\exists x\, y\, [\, [BLACK(x)\ \&\ SPHERE(y)] \Rightarrow [BLACK(A)\ \&\ SPHERE(A)]\,]$
we shall try to derive the empty clause from its negation -Th.
As usually [Kowalski 1979], the theorem is transformed into clauses as follows.
$-\exists x\, y\, [\, [BLACK(x)\ \&\ SPHERE(y)] \Rightarrow [BLACK(A)\ \&\ SPHERE(A)]\,] =$
$-\exists x\, y\, [\, -[BLACK(x)\ \&\ SPHERE(y)] \vee [BLACK(A)\ \&\ SPHERE(A)]\,] =$
$\forall x\, y\, [\, [BLACK(x)\ \&\ SPHERE(y)]\ \&\ -[BLACK(A)\ \&\ SPHERE(A)]\,] =$
the set of clauses :
BLACK(x) :-
SPHERE(y) :-
:- BLACK(A), SPHERE(A).

This set generates once only the empty clause with the substitution
$\sigma = \{ x<-A, y<-A \}$.
It follows that the set $\{ x=A, y=A \} = P_i$ is the set for which Th is valid.
As proven above, conjuncting -[$(x=A)$ & $(y=A)$] to [BLACK(x) & SPHERE(y)] will give the looked for generalization :

[BLACK(x) & SPHERE(y)] & [$(x \neq A) \vee (y \neq A)$].

This is the generalization which keeps as much as possible what has been deduced from the examples and excludes the counter-examples.
It shows that from one counter-example alone no too much subtle information can be deduced.

One should be aware that this is the best particularization that can be made from the counter-examples. If one attempted to derive a more general law, one could over-generalize and lose some vital information.

In case one adds the following further counter-example :

CE_7 = BLACK(A) & SPHERE(D)

then one must be able to deduce that the generalization from f_6 and CE_6 and CE_7 must be

[BLACK(x) & SPHERE(y) & $(x \neq A)$],

which is the result of our method.

Third case : Th is provable but does not reduce to the empty clause.

One must then analyse the failure. Since we suppose that we use resolution to deduce the empty clause from -Th, it follows that the failure will be caused by a subset of irreducible clauses that do not reduce to the empty clause Let us call LO(x) (Left-Over) this subset.
We shall not give here many details about LO(x), nor comment on the fact that it is not unique in general.
Section 4 is devoted to an analysis of this case.
Let us suppose in this section that it is unique.
Consider the expression $f(x)$ & -LO(x) and attempt to prove
Th' : $\exists x$ [$f(x)$ & -LO(x)] => $f_C(x)$]
Its negation is
$\forall x$ [[$f(x)$ & -LO(x)] & -$f_C(x)$] and, since LO(x) is precisely the left-over of the resolution of $\forall x$ [$f(x)$ & -$f_C(x)$], one will deduce the empty clause from it.
Let us now call P_i the set of values that verify Th'.

Drawing a conclusion from the above reasoning requires us to take a somewhat closer look at LO.
Due to the conjunctive form of the theorem $\forall x$ [$f(x)$ & -LO(x) & -$f_C(x)$], one can always assume that each clause contains atoms that originate from and only from $f(x)$ or from and only from $f_C(x)$. It follows that LO(x) has the form LO'(x) & -LO''(x) where

$f(x) = f'(x)$ & LO'(x) and $f_C(x) = f'_C(X)$ & LO''(x).

During the proof of Th', LO' (respectively -LO'') resolves some predicates of -f_C (respectively f).
It follows that from the proofs of $\exists x$ [[$f(x)$ & -LO(x)] => $f_C(x)$] one can deduce that
$\forall x$ [[$f'(x)$ & -[$x \in P_i$]] => -$f'_C(x)$] by the same reasoning as above.
Let us now use the two following trivialities

from A => C, deduce that A & B => C
from A => C, deduce that A & D => C $\vee$ D

in order to find that
$\forall x\ [\ [f(x)\ \&\ -[x \in P_i]\ \&\ -LO''(x)]\ \Rightarrow\ -f_C(x)\ ]$
which the interesting form we wanted to construct.

The final definition we can now give for **the best formula** that can be learned from formula $f(x)$ and counter-examples generalizing to $f_C(x)$ is:

$$[\,f(x)\ \&\ -LO''(x)\ \&\ -[x \in P_i]\,].$$

This is the correct recognition function.

We now describe two simple examples showing that our definitions contain the well-known intuitive learning behaviour when the examples and counter-examples mismatch by a predicate.

First case : the generalization from examples contains more predicates than the counter-example.

Recall that $f_6(x, y)$ is : BLACK(x) & SPHERE(y), and suppose that the counter-example is now BLACK(A).
The attempt to prove
$\exists x\, y\ [\ [BLACK(x)\ \&\ SPHERE(y)]\ \Rightarrow\ BLACK(A)]$
fails with SPHERE(y) as LO = LO'.
Conjuncting $-LO$ to $f_6(x, y)$, one attempts now to prove
$\exists x\, y\ [\ [BLACK(x)\ \&\ SPHERE(y)\ \&\ -SPHERE(y)]\ \Rightarrow\ BLACK(A)]$, *i.e that*
$-\exists x\, y\ [\ BLACK(x)\ \Rightarrow\ BLACK(A)\ ]$
contains a contradiction. The substitution $\{x \leftarrow A\}$ describes the domain where this contradiction holds, and it follows that
$\forall x\, y\ [\ [BLACK(x)\ \&\ SPHERE(y)\ \&\ -SPHERE(y)\ \&\ (x \neq A)]\ \Rightarrow\ -BLACK(A)]$.
This shows that the final generalization is
$$[BLACK(x)\ \&\ SPHERE(y)\ \&\ SPHERE(y)\ \&\ (x \neq A)] =$$
$$[BLACK(x)\ \&\ SPHERE(y)\ \&\ (x \neq A)]$$

Second case : the generalization from examples contains less predicates than the counter-example.

Let the generalization from examples be $f_7(x)$ = BLACK(x), and the counter-example be SPHERE(A) & BLACK(A).
The left-over is $-SPHERE(A) = -LO''$, conjuncting its negation to BLACK(x) allows us to find the empty clause with $\{x \leftarrow A\}$
It follows that the best recognition function is
$$BLACK(x)\ \&\ -SPHERE(A)\ \&\ (x \neq A)$$

4 - IMPROVING THE EXPLANATIONS

4.1 - Improving the quality of the generalization

Instead of applying the above techniques to the recognition function and a counter-example, one can also attempt to compare it to a generalization of the

counter-examples.
Since their generalization will be used in order to expell some information from the recognition function of the examples, it mav be that the modified recognition function no longuer recognizes all the examp..es, after its modification by an over-generalization of the ples.
It is therefore extremely important to avoid over-generalizing the counter-examples.
Structural Matching, the role of which is to avoid such kind of overgneralization, is important when generalizing examples, but it is even more important when generalizing counter-examples.

Consider again
f_8 = *BLACK(x) & SPHERE(y).*
Consider now the case where one wants to find the correct generalization associated to f_8 *and*

$$CE_8 = BLACK(A) \ \& \ SPHERE(A)$$
$$CE_9 = BLACK(B) \ \& \ SPHERE(B).$$

The recognition function deduced from CE_8 *and* CE_9 *is*
$f_C(x) = BLACK(x) \ \& \ SPHERE(x)$

In this case, Th is

$$\exists x\ y\ [\ [BLACK(x) \ \& \ SPHERE(y)] => [BLACK(x) \ \& \ SPHERE(x)]\ ]$$

which is TRUE for the unique substitution $\{y<-x\}$, *therefore* P_i *is characterized by* $x = y$ *and* $\neg[x \in P_i]$ *if* $x \neq y$. *It follows that the correct generalization is, in this case,*

$$[BLACK(x) \ \& \ SPHERE(y) \ \& \ (x \neq y)].$$

On can notice that over-generalizing CE_1 *and* CE_3 *to BLACK(x) & SPHERE(y) for instance would lead to total disappearance of the recognition function, a case clearly difficult to overcome by further modifications !*

4.2 - Improvment of the quality of the proof

As seen in preceding sections, an explanation procedure can always be attached to a proof of recognition or rejection.
There are often several possible proofs. Each of them will provide new different explanations.
Explanations relative to successes will enrich the recognition functions.
Explanations relative to failures will allow modifying the data basis. For instance, in the example of section 3.3, the default values of an *ENDTABLE* can be modified in case the system fails to recognize that a given BOX_1 can be stacked on a given $ENDTABLE_1$.
In a real situation, where we will have to handle numerous explanations, one will be able to modify the data theory by learning.
The following ad'hoc examples will illustrate the problems met during this process.

4.2.1 - The system generates several explanations from one example

Suppose that, as in section 3.3, it finds that BOX_1 can be stacked on $ENDTABLE_1$ because the weight of BOX_1 is less than 50. Suppose also that, by using an other reasoning path, it finds also the other explanation : because $ENDTABLE_1$ is "very stiff" (one should have defined this predicate in the data of the theory).
The system will have first to prove that there is no mutual implication between

"BOX_1 is less than 50" and "$ENDTABLE_1$ is very stiff".
Then it will have to improve its explanation by providing a disjunction of these two cases.

This very simple example already shows that the proofs of relations among different explanations must be carefully studied.

As an other example, suppose that one obtains the two explanations
Ex1 : the weight of BOX_1 is less than 50
Ex2 : the weight of $ENDTABLE_1$ is more than twice the weight of BOX_1
depending on the reasoning path that is used during the proof.

In this case, one would have to check that the weight of BOX_1 is less than 25, i.e. that the two explantions are not contradicting each other.
None of them imply the other, they would have both to be kept.

4.2.2 The system generates different explanations for different examples

One has also to check the logical dependency of the explanations.

Suppose that comparing different examples, the system is able to recognize that for each example E_i the ratio : weight of $ENDTABLE_i$ / weight of BOX_i is almost constant.
This remark would favor explanations like Ex2, where a ratio is involved.

This is a typical case where EBL and SBL should concur : one has to use SBL-like methods on data obtained by EBL-like methods.

5 - CONCLUSION

A first conclusion is relative to the importance of formal proofs of the validity of the obtained recognition function. They allow to add or delete the exact amount of information which is needed to be added or deleted.

A second conclusion is relative to the way the explanation abilities can be improved by theorem proving. There are to ways to achieve this goal.
The first one is "first order", i.e., one improves the quality of each proof, due to a progressive refinement of the data basis.
The second one is "second order", i.e., one improves the quality of the combinations of the different explanations issued from different proofs.

REFERENCES

[Buchanan & Al. 1971] Buchanan B.G., Feigenbaum E.A., Lederberg J. "A heuristic programming study of theory formation in sciences", Proceedings of the Second International Joint Conference on Artificial Intelligence, Londres 1971, pages 40-48.

[Buchanan & Mitchell 1978] Buchanan B.G., Mitchell T.M. "Model-directed learning of production rules", in *Pattern-directed inference systems*, Waterman D.A. and Hayes-Roth F. eds., Academic Press, New York 1978.

[Brazdil 1978] P. Brazdil, "Experimental Learning Model", Proc. 3rd AISB meeting, Hamburg 1978, pp. 46-50.

[Bundy & Al. 84] Bundy A., Silver B., Plummer D. "An Analytical Comparison of Some Rule Learning Programs", Univ. Edinburgh, DAI Res. paper 125, 1984.

[Cohen & Sammut 1984] Cohen B., Sammut C.: "Program synthesis through concept learning", in *Automatic Program Construction Techniques*, Biermann A.W., Guiho G., Kodratoff Y. eds, Macmillan Publishing Company,1984, pp. 517-552. Macmillan Publishing Company,1984, pp. 463-482.

[Costa 1982] E. J. F. Costa "Dérécursivation automatique en utilisant des systèmes de réécriture de termes", Thèse, Paris 1982. Publication Interne LRI 118.

[Dejong 1981] Dejong G., "Generalizations Based on Explanations", Proc. 7th IJCAI, 1981, pp. 67-69.

[Dietterich 81] Dietterich G.T., Michalski R.S.: "Inductive learning of structural descriptions : Evaluation criteria and comparative review of selected methods" Artificial Intelligence Journal 16, 1981, 257-294.

[Ganascia 85] Ganascia J.G. "Comment oublier à l'aide de contre-exemples?" Actes du congrès AFCET RFIA, Grenoble, Novembre 1985.

[Hayes-Roth 78] Hayes-Roth F., McDermott J.: "An interference matching technique for inducing abstractions", C. ACM 21, 1978, 401-411.

[Kodratoff 1983] Kodratoff Y., "Generalizing and Particularizing as the Techniques of Learning", Computers and Artificial Intelligence 2, 1983, 417-441.

[Kodratoff & Al. 1984] Kodratoff Y., Ganascia J.-G., Clavieras B., Bollinger T., Tecuci G., "Careful generalization for concept learning" Proc. ECAI-84, Pisa 1984, pp. 483-492. Now also available in *Advances in Artificial Intelligence*, T. O'Shea editor, pp. 229 - 238, North - Holland Amsterdam 1985.

[Kodratoff & Duval 1986] Kodratoff Y., Duval B., "???", Proc ECAI - 86

[Kodratoff & Ganascia 1986] Kodratoff Y., Ganascia J.-G., "Improving the Generalization Step in Learning", in *Machine Learning, An Artificial Intelligence Approach, Volume 2*, Michalski, R.S., Carbonell, J. G., Mitchell, T.M. (eds), Morgan-Kaufmann 1986, pp. 215-244.

[Kowalski 1979] Kowalski R. *Logic for Problem Solving*, North Holland 1979.

[Langley 1983] P.Langley, "Learning Search Strategies through discrimination", Int. J. Man-Machine Studies 18, 1983, 513-541.

[Lebowitz 1986] Lebowitz M., "Integrated Learning : Controlling Explanation", *Cognitive Science* 10, 1986, to appear.

[Michalski & Chilauski 1980] Michalski R. M., Chilauski R. L. "Learning by Being Told and Learning from Examples : An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis", Internatl. J. of Policy Analysis and Information Systems 4, 1980.

[Michalski & Al. 1982] Michalski R. M., Davis J. H., Bisht V. S., Sinclair J. B. "PLANT/ds : An Expert Consulting System for the Diagnostic of Soybean Diseases", Proc. ECAI-82, Orsay 1982, pp. 133-138.

[Michalski & Stepp 1983] Michalski R. S., Stepp R. E. "Learning from Observation : Conceptual Clustering", in *Machine Learning, an Artificial Intelligence Approach*, Michalski R.S., Carbonell J.G., Mitchell T.M. eds, Tioga Publishing Company 1983, pp. 163-190.

[Michalski 1984] Michalski R.S., "Inductive Learning as Rule-guided Transformation of Symbolic Descriptions : a Theory and Implementation", in *Automatic Program Construction Techniques*, Biermann A.W., Guiho G., Kodratoff Y. eds, Macmillan Publishing Company, 1984, pp 517-552.

[Mitchell 83] Mitchell T.M.: "Learning and Problem Solving" Proc. IJCAI-83, Karlsruhe 1983, pp. 1139-1151.

[Mitchell, Utgoff & Banerji 1983] Mitchell T.M., Utgoff P.E., Banerji R. "Learning by experimentation, acquiring and refining problem-solving heuristics", in *Machine Learning, an Artificial Intelligence Approach*, Michalski R.S., Carbonell J.G., Mitchell T.M. eds, Tioga Publishing Company 1983, pp. 163-190.

[Mitchell 1985] Mitchell T. M., Mahadevan S., Steinberg L. I., "Leap : A Learning Apprentice for VLSI Design", Proc. IJCAI-85, Los Angeles 1985, pp. 573-580.

[Nilsson 1980] Nilsson N. J. *Principles of Artificial Intelligence*, Tioga Pub. Comp. 1980.

[Porto 1983] Porto A., "Logical Action Systems", Proc. Logic Programming Workshop'83, Portugal July 1983, pp. 192 - 203.

[Quinlan 1983] Quinlan J. R., "Learning Efficient Classification Procedures and their Application to Chess End Games" in *Machine Learning, An Artificial Intelligence Approach*, Michalski, R.S., Carbonell, J. G., Mitchell, T.M. (Eds.), Tioga Publishing Company, 1983, pp. 463-482.

[Rosenblatt 1958] Rosenblatt F. "The perceptron : A probabilistic model for information storage and organization in the brain", Psychological Review 65, 1958, 386-407.

[Samuel 1959, 1963] Samuel A.L. "Some studies in Machine Learning using the game of checkers", IBM Journal of Research and Development 3, 1959, 211-229.
Samuel A.L. " Some studies in Machine Learning using the game of checkers", in *Computer and Thought*, Feigenbaum E.A. et Feldman J. editeurs, McGraw-Hill New-York 1963, pp 71-105.

[Silver 1983] Silver B., "Precondition Analysis : Learning Control Information", in *Machine Learning, An Artificial Intelligence Approach, Volume 2*, Michalski R.

S., Carbonell J. G., Mitchell T. M eds. Morgan Kaufmann, Los Altos 1986, pp 647 - 670.

[Touretzky & Hinton 1985] Touretzky D. S., Hinton G. E. "Connectionist Inference Architecture, Proc IJCAI-85, Los Angeles, 1985, pp 238-243.

[Vere 80] Vere S.A.: "Multilevel counterfactuals for generalizations of relational concepts and productions" Artificial Intelligence J. 14, 1980 139-164.

[Vere 81] Vere, S.A., "Constrained N-to-1 Generalizations", unpublished draft, 23. Feb, 1981.

[Vrain 85] Vrain C. "Contre-exemples : explications déduites de l'étude des prédicats", Actes congrès AFCET RF-IA, Grenoble 1985, pp 145 - 159.

[Waldinger 1977] Waldinger R. "Achieving Several Goals Simultaneously", in *Machine Intelligence 8*, E. W. Elcock and D. Michie Eds, Ellis Horwood 1977.

[Winston 1975] Winston P. H., "Learning Structural Descriptions from Examples", in *The Psychology of Computer Vision*, Winston P.H. (eds), Ch. 5, McGraw Hill 1975.

ADDUNDUM

TO PROCEEDINGS OF THE INTERNATIONAL MEETING ON ADVANCES IN LEARNING

<u>IMAL ~ 1986</u>

Les Arcs, July 28th - August 1st 1986

Not the Path to Perdition: The Utility of Similarity-Based Learning

Michael Lebowitz[1]
Department of Computer Science -- Columbia University
New York, NY 10027

27 June 1986

Abstract

A large portion of the research in machine learning has involved a paradigm of comparing many examples and analyzing them in terms of similarities and differences, assuming that the resulting generalizations will have applicability to new examples. While such research has been very successful, it is by no means obvious why similarity-based generalizations should be useful, since they may simply reflect coincidences. Proponents of explanation-based learning, a new, knowledge-intensive method of examining single examples to derive generalizations based on underlying causal models, could contend that their methods are more fundamentally grounded, and that there is no need to look for similarities across examples. In this paper, we present the issues, and then show why similarity-based methods *are* important. We present four reasons why robust machine learning must involve the integration of similarity-based and explanation-based methods. We argue that: 1) it may not always be practical or even possible to determine a causal explanation; 2) similarity usually implies causality; 3) similarity-based generalizations can be refined over time; 4) similarity-based and explanation-based methods complement each other in important ways.

Topics: Knowledge acquisition and learning; concept learning; explanation-based learning

1 Introduction

Until recently, machine learning has focused upon a single paradigm -- the generalization of concepts through the comparison of examples. The assumption has been made, though often tacitly, that the generalization of similarities will lead to concepts that can be applied in other contexts. Despite its ubiquity there is one real problem with this paradigm: there is no obvious reason why the underlying assumption should hold. In other fields people have called into doubt the utility of noticing similarities in the world and assuming them to be important. Naturalist Stephen Jay Gould, in discussing the nature of scientific discovery comments that:

The human mind delights in finding pattern -- so much so that we often mistake coincidence or

[1]This research was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0165 and in part by the United States Army Research Institute under contract MDA903-85-0103. Comments by Kathy McKeown on an earlier draft of this paper were quite useful.

NB - This paper has been already issued in the AAAI - 86 Proceedings. Therefore, it will be subject to deep modifications before being publicly re-issued.

forced analogy for profound meaning. No other habit of thought lies so deeply within the soul of a small creature trying to make sense of a complex world not constructed for it.

'Into this Universe, and why not knowing // Nor whence, like water willy-nilly flowing' as the *Rubaiyat* says. No other habit of thought stands so doggedly in the way of any forthright attempt to understand some of the world's most essential aspects -- the tortuous paths of history, the unpredictability of complex systems, and the lack of causal connection among events superficially similar.

Numerical coincidence is a common path to intellectual perdition in our quest for meaning. [Gould 84]

Further doubt has been cast upon the use of similarity-based learning by a new methodology that has been developed in the last few years: the extensive application of knowledge to single examples to determine the underlying mechanism behind an example, and the use of this causal explanation to derive generalized concepts. By learning from single examples, this knowledge-based approach calls into question the necessity of similarity-based approaches.

Despite Gould's warning and the recent successes of explanation-based methods, learning methods that concentrate on seeking out coincidences have had remarkable success across a variety of tasks. Furthermore, as Gould implies above, people (and other creatures) do seem to be optimized for such learning. Given this evidence, it worth trying to explain why such methods work. In this paper we will explain why similarity-based learning not only works, but is a crucial part of learning.

2 EBL and SBL

Considerable research has been done involving *similarity-based learning* (SBL). [Winston 72; Winston 80; Michalski 80; Michalski 83; Dietterich and Michalski 86; Lebowitz 83; Lebowitz 86a] are just a few examples. (See also, [Michalski et al. 83; Michalski et al. 86].) While there are many variations to such learning research, the basic idea is that a program takes a number of examples, compares them in terms of similarities and differences, and creates a generalized description by abstracting out similarities. A program given descriptions of Columbia University and Yale University and told that they were Ivy League universities and that the University of Massachusetts was not would define "Ivy League university" in terms of the properties that the first two examples had and that the third did not -- e.g., as being private, expensive and old. Similarity-based learning has been studied for cases where the input is specially prepared by a teacher; for unprepared input; where there are only positive examples; where there are both positive and negative examples; for a few examples; for many examples; for determining only a single concept at a time; and for determining multiple concepts. In a practical sense, SBL programs have learned by comparing examples more or less syntactically, using little "high level" knowledge of their domains (other than in deciding how to represent each example initially).

Explanation-based learning (EBL), in contrast, views learning as a knowledge-intensive activity, much like other tasks in Artificial Intelligence. [DeJong 86; Ellman 85; Mitchell 83a; Mostow 83; Minton 84; Silver 86] are a few examples of explanation-based learning research. (See also [Michalski et al. 86].) An EBL program takes a single example, builds up an explanation of how the various components relate to each other at a low level of detail by using traditional AI understanding or planning methods, and then generalizes the properties of various components of the example so long as the explanation remains valid. What is left is then viewed as a generalized description of the example that can be applied in understanding further examples. This kind of learning is tremendously useful, as it allows generalized concepts to be determined on the basis of a single example. On the other hand, the building and analysis of explanations does require extremely detailed knowledge of the domain (which may minimize the need to learn). In addition, virtually all current EBL work is in the "perfect learner" paradigm that assumes that all input is noise-free and fits the correct final generalization.

It is important to make clear here exactly the sense in which EBL is concept learning. It might be contended that all that is being done is the application of pre-existing information to a problem, unlike SBL, which is clearly a form of inductive learning. The key is in the generalization phase, where the EBL learner loosens constraints on its representation and determines whether the explanation that it has built up still holds. This generalized concept can then serve as a form of compiled knowledge that simplifies the processing of later input. This may be a way to learn structures such as frames [Minsky 75] and scripts [Schank and Abelson 77]. The view of using EBL to produce knowledge structures that make later processing more efficient has been called *operationalization* [Mostow 83]. Even though it might in some sense be possible to understand later examples just using low-level rules, realistically it is crucial to have a set of knowledge structures at various levels of complexity.

3 The goal of learning

It does not make sense to consider learning in isolation from other elements of intelligent processing. While certain aspects of learning may not be in service of an immediate goal (e.g., curiosity), at some point there must be a task involved to make use of what is learned. In general, the idea is for an organism or program to be able to carry out a task better (either be able to do more examples or do examples more efficiently) than it did before learning. It is particularly important to keep in mind the task nature of learning when considering concept learning, which has often been studied without regard to the future utility of the concepts created.

For most tasks that people or intelligent programs will carry out, the most obvious way to be able to improve performance is to attempt to develop a *causal model* that explains how elements of the domain work. Such a model will allow the learner to *predict* what is likely to happen in later situations, which will clearly be useful. The model will allow the learner to *understand* further input. Although we will consider later whether it is possible in all domains, the construction of a causal model is clearly a worthy goal in learning. [Schank 75; Schank 84] present reasons for constructing such models even in domains with incomplete models. Explanation-based learning methods strike directly at the problem of creating causal models. Similarity-based methods do not, but yet seem to lead to useful generalizations. This leads us to the central mystery of this paper.

4 The puzzle

Having decided that the construction of a causal model for a domain is important, or perhaps even crucial, as part of learning, we are left with the key question, "Is there any role for similarity-based learning in a full learning model, and if so, why?" Even if we assume that there must be something to SBL, since, after all, so many people have worked on it with impressive results, we must ask why it works; why it helps a learner perform better. That generalizations from explanation-based learning are valid and useful makes sense intuitively, since they are derived from causal analyses. Similarity-based generalizations could just be the result of the coincidences that arise in a complex world.

Note that similarity-based learning is not merely an artifact of researchers in machine learning. As pointed out in the Gould quote above, people delight in noticing similarities in disparate situations. Indeed, in many ways human processing seems to be optimized for such learning. An anecdotal example immediately comes to mind: On the Eastern Air Shuttle between New York and Boston, passengers are given a sequence number for boarding. On one roundtrip, I received the same sequence number going in each direction. I noticed the similarity immediately, even though the first number was not in front of me when I received the second, despite the apparent irrelevance of the coincidence to my performance on later shuttle trips. Virtually everyone has experienced, and noticed, similar coincidences. When nature provides such a powerful cognitive mechanism, there always seems to be a good reason. We will see shortly why the recognition of similarities is important, though, to reiterate, the utility is *not* obvious and should not simply be assumed by SBL researchers.

5 A similarity-based learning program

We can most easily look at the utility of SBL in the context of a specific learning program. UNIMEM [Lebowitz 82; Lebowitz 86a; Lebowitz 86b] takes examples represented as sets of features (essentially property/value pairs) and automatically builds up a generalization hierarchy using similarity-based methods. It is not told in advance which examples to compare or concepts to form, but instead learns by observation. One domain on which we have tested UNIMEM involves data about universities that was collected from students in an Artificial Intelligence class at Columbia.[2]

Figure 1 shows the information used by UNIMEM for two universities, Columbia and Carnegie-Mellon. Each university is represented by a set of triples that describe features of the university, the first two providing a property name and the third its value. So, Columbia is in New York State while Carnegie-Mellon is in Pennsylvania. Both are urban and private and Columbia has a 7/3 male/female ratio compared to Carnegie-Mellon's 6/4. Some features, like quality of life, involve arbitrary numeric scales.

FEATURE:		COLUMBIA:	CMU:
STATE	VALUE	NEW-YORK	PENNSYLVANIA
LOCATION	VALUE	URBAN	URBAN
CONTROL	VALUE	PRIVATE	PRIVATE
MALE:FEMALE	VALUE	RATIO:7:3	RATIO:6:4
NO-OF-STUDENTS	VALUE	THOUS:5-	THOUS:5-
STUDENT:FACULTY	VALUE	RATIO:9:1	RATIO:10:1
SAT	VERBAL	625	600
	MATH	650	650
EXPENSES	VALUE	THOUS$:10+	THOUS$:10+
%-FINANCIAL-AID	VALUE	60	70
NO-APPLICANTS	VALUE	THOUS:4-7	THOUS:4-7
%-ADMITTANCE	VALUE	30	40
%-ENROLLED	VALUE	50	50
ACADEMICS	SCALE:1-5	5	4
SOCIAL	SCALE:1-5	3	3
QUALITY-OF-LIFE	SCALE:1-5	3	3
ACAD-EMPHASIS	VALUE	LIB-ARTS	ENGINEERING

Figure 1: Information about two universities

The first question we have to address concerning the examples in Figure 1 is precisely what it means to "understand" them, or to learn from them. While the exact nature of understanding would depend on the ultimate task that we had in mind, presumably what a person or system learning from

[2]Other domains UNIMEM has been tested on include: information about states of the United States, Congressional voting records, software evaluations, biological data, football plays, universities, and terrorism stories.

these examples would be after is a causal model that relates the various features to each other.

As an example, in understanding Figure 1 we might wish to know how the fact that both universities are private relates to the fact that they are both expensive or why Carnegie-Mellon offers financial aid to more people. A causal model that answers questions of this sort would be extremely useful for almost any task involving universities. Typical of the causation that we would look for is, for example, that private universities get less government support and hence have to raise more money through tuition. (At least that is how private universities explain it!) Similarly, a model might indicate that Carnegie-Mellon's emphasis on engineering leads to the acceptance of more students who need financial aid. Notice, however, that it will certainly not be possible to build a complete causal model solely from the information in Figure 1, but will require additional domain knowledge.

An EBL program would create a low-level causal model of a university using whatever methods were available and then would use the model to develop a generalized concept. For example, it might decide that the Columbia explanation could be generalized by removing the requirement of being in New York State and by allowing the numeric values to vary within ranges, if none of these changes would affect the underlying explanation. It might be, however, that the liberal arts emphasis is crucial for some aspect of the explanation. In any case, by relaxing constraints in the representation, an EBL program would develop, using a single, causally motivated example, a generalized concept that ought to apply to a wide range of situations.

Let us now compare the desired causal explanation with the kind of generalization made using similarity-based methods. Figure 2 shows the generalization that is made by UNIMEM, GND1, from the two university representations in Figure 1.[3] We see in Figure 2 that UNIMEM has generalized Columbia and Carnegie-Mellon by retaining the features that have identical values (like social level and quality of life), averaging feature values that are close (such as SAT verbal score) and eliminating features that are substantially different, such as the state where the university is located and the percentage of financial aid.[4] The resulting set of features can be viewed as a generalization of the two examples, as it describes both of them, as well as, presumably, other universities that differ in other features.

[3]Actually, UNIMEM also had to decide that these two examples should even be compared and that they had a substantial amount in common before doing the actual generalization.

[4]Exactly what constitutes "substantially different" is a parameter of the program.

```
GND1
    SOCIAL               SCALE:1-5    3
    QUALITY-OF-LIFE      SCALE:1-5    3
    LOCATION             VALUE        URBAN
    CONTROL              VALUE        PRIVATE
    NO-OF-STUDENTS       VALUE        THOUS:5-
    STUDENT:FACULTY      VALUE        RATIO:9:1
    SAT                  MATH         650
    SAT                  VERBAL       612.5
    EXPENSES             VALUE        THOUS$:10+
    NO-APPLICANTS        VALUE        THOUS:4-7
    %-ENROLLED           VALUE        50
    [CARNEGIE-MELLON COLUMBIA]
```

Figure 2: Generalizing Columbia and Carnegie-Mellon

What would the generalization in Figure 2 be used for once it had been made? Presumably it would be used in processing information about other universities. If we identified a situation where GND1 was thought to be relevant, we would assume that any of its features that were not known would indeed be present. The assumption is made by all similarity-based learning programs, including UNIMEM, that they have created usable concepts from which default values may be inherited.

We can now state our problem quite clearly in terms of this example: *What reason do we have to believe that a new example that fits part of the generalization of Columbia and Carnegie-Mellon will fit the rest?* With explanation-based methods we at least have the underlying causal model as justification for believing the generalization. But what is the support of similarity-based learning?

6 Elements of an answer

There are four main elements to our explanation as to why SBL produces generalized concepts that can be profitably applied to other problems and why it should be so used:

- While the goal of learning is indeed a causal model, it is often not possible to determine underlying causality and even where it is possible it may not be practical.
- Similarity usually implies causality and is much easier to determine.
- There are ways to *refine* generalizations to mitigate the effects of coincidence.
- Explanation-based and similarity-based methods complement each other in crucial ways.

6.1 Causality cannot always be determined

In order to achieve their impressive results, the EBL methods that have been developed to date assume that a complete model of a domain is available and thus a full causal explanation can be constructed. In addition, it is assumed that it is always computationally feasible to determine the explanation of any given example. While these assumptions may be acceptable for some learning tasks, they do not appear reasonable for situations where we are dealing with noisy, complex, uncertain data -- characteristics of most real-world problems. It is also unreasonable to expect to have a complete domain model available for a new domain that we are just beginning to explore. Even in our university example, it is hard imagine all the information being available to build a complete model.

Most EBL work has not addressed these issues. Some of the domains used, like integration problems [Mitchell 83a], logic circuits [Mitchell 83b; Ellman 85] or chess games [Minton 84] do indeed have complete domain models and the examples used are small enough for the explanation construction to be tractable. Even in a domain such as the news stories of [DeJong 86], the assumption is made, perhaps less validly, that it is always possible to build up a complete explanation.

In domains where a detailed explanation cannot reasonably be constructed, a learner can only rely on similarity-based methods. By looking for similarities it is at least possible for the learner to bring some regularity to its knowledge base. The noticing of co-occurrence is possible even the absence of a complete domain model. Further, much research, including our own, has shown that SBL can be done efficiently in a variety of different problem situations. In the university example of Section 5, UNIMEM was able to come up with a variety of similarity-based generalizations with minimal domain information. Further, as we noted above, people seem to be optimized for SBL.

6.2 Similarity usually implies causality

The regularity that is detected using SBL is not worthwhile if it cannot be used to help cope with further examples. Such help is not likely if there is no connection between the similarities and the underlying causal explanation. Fortunately, such a connection will usually exist.

Put as simply as is possible, similarities among examples usually occur because of some underlying causal mechanism. Clearly if there is a consistent mechanism, it will produce consistent results that can be observed as similarities. While the infinite variety of the world will also produce many coincidental similarities, it is nonetheless true that among the observed similarities are the mechanisms that we desire.

So, in the Eastern Shuttle example used above, while it is almost certain that the duplicate seat numbers I received were coincidental, if there *was* a mechanism involving seat numbers (say the numbers were distributed in alphabetical order) it would manifest itself in this sort of coincidence. Similarly, in the university generalization GND1 (Figure 2), we indicated possible of mechanisms that would lead to the kind of expensive private school that is described.

Two recent examples illustrate how causal understanding frequently relates to similarity-based processing. The first involves scientific research, an attempt to understand a complex meteorological phenomenon, and the second an investigation into a mysterious crime.

In recent years weather researchers have been trying to explain a set of possibly *related facts*. Specifically: 1) the average temperature in 1981 was very high; 2) the El Chichon volcano erupted spectacularly in early 1982; 3) El Nino (a warm Pacific current) lasted an exceptionally long time starting in mid-1982; 4) there have been severe droughts in Africa since 1982.

One might expect researchers to immediately attempt to construct a causal model that explains all these phenomena. However, weather systems are extremely complex, and by no means fully understood. Author Gordon Williams, writing in *Atlantic*, discusses the attempt to gain understanding as follows: "How could so much human misery in Africa be caused by an errant current in the Pacific? *Records going back more than a century show that the worst African droughts often come in El Nino years.*" (Emphasis added.) Furthermore, Williams quotes climate analyst Eugene Rasmusson as saying, "It's disturbing because we don't understand the process" [Williams 86].

We can see clearly in this example that although the ultimate learning goal is a causal model, the construction of such a model is not immediately possible. So, researchers began by looking for correlations. However, they expect correlations to lead eventually to deeper understanding.

The second example involves investigators trying to determine how certain extra-strength Tylenol capsules became laced with poison. The *New York Times* of February 16, 1936 reported:

> Investigators tracing the routes of two bottles of Extra-Strength Tylenol containing cyanide-laced capsules have found that both were handled at the same distribution center in Pennsylvania two weeks apart last summer. Federal officials and the product's manufacturer said that the chance that the tainting occurred at the distribution facility was remote, but the finding prompted investigators to examine the possibility as part of their inquiry." [McFadden 86]

Again we have a case where a causal explanation is desired and yet there is not enough information available to construct one. So, the investigators began by looking for commonalities among the various poisoned capsules. When they found the distribution facility in common, that became an immediate possible *contributor* to the explanation. Although no final explanation had been discovered as this is written, it is clear that the explanation process attempted began with the noticing of similarities.

There is one further connection between noticing similarities and generating explanations that is worth making. This involves the idea of *predictability*. It turns out that the kinds of similarities that are noticed provide clues not only to what features should be involved in an explanation, but what the direction of causality might be (e.g., what causes what). As we have described elsewhere [Lebowitz 83; Lebowitz 86c], features that appear in just a few generalizations, which we call *predictive*, are the only ones that indicate a generalization's relevance to a given situation, and, further, are those likely to be the *causes* in an underlying explanation. This becomes clear when we realize that a feature present in many different situations cannot cause the other features in any single generalization, or it would cause the same features to appear in *all* the other generalizations that it is in.

In the weather example above, if we knew of many generalizations involving droughts, but only one with both warm currents and a volcano, then the volcano might cause the drought, but the drought could not cause the volcano. Of course, it may be that neither direction of causality is right, there being a common cause of both, but at least predictability provides a starting point.

The power of predictability is that it can be determined quite simply, basically as a byproduct of the normal SBL process. The various indexing schemes used in a generalization-based memory [Lebowitz 83; Lebowitz 86a] allow the simple counting of features in context. While there are many problems to be explored, particularly that of predictive combinations of features, the ability to know the likely initial causes when determining a mechanism is an important advantage of SBL. Further, even when no explanation can be found, the use of predictability often allows us to make predictions from a generalization at the correct moments, even without any deep understanding of the generalization

6.3 Refining generalizations

The third part of our explanation as to the utility of similarity-based learning is that generalizations, once made, are not immutable – they can be *refined* in the light of later information. This means that the aspects of a generalizations that are due to coincidence can be removed. We have developed various techniques for doing this [Lebowitz 82] that work essentially by noticing during processing when various elements of a generalization are contradicted by new examples. If we remove the features that are frequently contradicted we can have a concept that is more widely applicable and contain meaningful information.

As an example of this, we will look again at our university generalization (Figure 2). Suppose that there were a wide range of universities with most of the features of GND1, but with different levels of social life. This contradiction of the social level value that was derived from the coincidental value that both Columbia and Carnegie-Mellon have might seem to invalidate the generalization. However, our refinement methods would allow UNIMEM (or a similar system) to remove this feature, leaving a more widely applicable generalization that describes high-quality private schools. In this way similarity-based methods can overcome some of the coincidences that might seem to require explanation-based methods. Notice, however, that UNIMEM makes this refinement without having any real idea of why it is doing so, other than the pragmatic rationale that it allows the generalization to fit more examples, but does not reduce it so much that it carries no information.

6.4 Integrated learning

The final element of our explanation for the importance of similarity-based methods lies in the need for an integrated approach employing both similarity-based and explanation-based approaches. This point is really a corollary of the relation between similarity and causality described in Section 6.2.

The basic idea is to use EBL primarily upon the generalizations that are found using SBL rather than trying to explain everything in sight. This drastically cuts down the search necessary for constructing an explanation, particularly in domains where we have very little specific knowledge and have to rely on general rules for the explanations. Basically, we use SBL as a bottom-up control on the top-down processing of EBL.

The "real world" weather and crime investigation examples in Section 6.2 illustrate clearly how human problem solvers make use of this form of integrated learning -- trying to explain the coincidences that are noted, rather than explaining every element of a situation from scratch. We have described how a simple form of such integrated learning has been implemented for UNIMEM in [Lebowitz 86c]. For the university example in Figure 5, the main point is that we would only try to build up an explanation for the generalization GND1 (actually, the version of GND1 refined over time), and not the specific examples that made it up. Explaining the generalization is likely to be much easier than explaining the features of Columbia and Carnegie-Mellon and provide almost as much information.

7 Conclusion

We have shown in this paper a number of ways that similarity-based learning can contribute to the ultimate learning goal of building a coherent causal explanation of a situation. From this analysis it is not surprising that people seem to be optimized for noticing similarities, as such processing leads to the understanding that helps deal with the world. Our computer programs should be equally well equipped. Similarity-based learning is definitely not the path to perdition.

References

[DeJong 86] DeJong, G. F. An approach to learning from observation. In R. S. Michalski, J. G. Carbonell and T. M. Mitchell, Ed., *Machine Learning: An Artificial Intelligence Approach, Volume II*, Morgan Kaufmann, Los Altos, CA, 1986, pp. 571 - 590.

[Dietterich and Michalski 86] Dietterich, T. G. and Michalski, R. S. Learning to predict sequences. In R. S. Michalski, J. G. Carbonell and T. M. Mitchell, Ed., *Machine Learning: An Artificial Intelligence Approach, Volume II*, Morgan Kaufmann, Los Altos, CA, 1986, pp. 63 - 106.

[Ellman 85] Ellman, T. Generalizing logic circuit designs by analyzing proofs of correctness. Proceedings of the Ninth International Joint Conference on Artificial Intelligence, Los Angeles, 1985, pp. 643 - 646.

[Gould 84] Gould, S. J. "The rule of five." *Natural History 93*, 10, October 1984, pp. 14 - 23.

[Lebowitz 82] Lebowitz, M. "Correcting erroneous generalizations." *Cognition and Brain Theory 5*, 4, 1982, pp. 367 - 381.

[Lebowitz 83] Lebowitz, M. "Generalization from natural language text." *Cognitive Science 7*, 1, 1983, pp. 1 - 40.

[Lebowitz 86a] Lebowitz, M. Concept learning in a rich input domain: Generalization-Based Memory. In R. S. Michalski, J. G. Carbonell and T. M. Mitchell, Ed., *Machine Learning: An Artificial Intelligence Approach, Volume II*, Morgan Kaufmann, Los Altos, CA, 1986, pp. 193 - 214.

[Lebowitz 86b] Lebowitz, M. UNIMEM, a general learning system: An overview. Proceedings of ECAI-86, Brighton, England, 1986.

[Lebowitz 86c] Lebowitz, M. "Integrated learning: Controlling explanation." *Cognitive Science 10*, 2, 1986, pp. 219 - 240.

[McFadden 86] McFadden, R. "Two bottles of poisoned tylenol were shipped by the same distributor." *New York Times 135*, February 16, 1986, pp. 1.

[Michalski 80] Michalski, R. S. "Pattern recognition as rule-guided inductive inference." *IEEE Transactions on Pattern Analysis and Machine Intelligence 2*, 4, 1980, pp. 349 - 361.

[Michalski 83] Michalski, R. S. "A theory and methodology of inductive learning." *Artificial Intelligence 20*, 1983, pp. 111 - 161.

[Michalski et al. 83] Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. (Eds.). *Machine Learning, An Artificial Intelligence Approach.* Morgan Kaufmann, Los Altos, CA, 1983.

[Michalski et al. 86] Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. (Eds.). *Machine Learning, An Artificial Intelligence Approach, Volume II.* Morgan Kaufmann, Los Altos, CA, 1986.

[Minsky 75] Minsky, M. A framework for representing knowledge. In P. H. Winston, Ed., *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975.

[Minton 84] Minton, S. Constraint-based generalization. Proceedings of the Fourth National Conference on Artificial Intelligence, Austin, TX, 1984, pp. 251 - 254.

[Mitchell 83a] Mitchell, T. M. Learning and problem solving. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, 1983, pp. 1139 - 1151.

[Mitchell 83b] Mitchell, T. M. An intelligent aid for circuit redesign. Proceedings of the Third National Conference on Artificial Intelligence, Washington, DC, 1983, pp. 274 - 278.

[Mostow 83] Mostow, J. Operationalizing advice: A problem-solving model. Proceedings of the 1983 International Machine Learning Workshop, Champaign-Urbana, Illinois, 1983, pp. 110 - 116.

[Schank 75] Schank, R. C. The structure of episodes in memory. In D. Bobrow and A. Collins, Ed., *Representation and Understanding: Studies in Cognitive Science*, Academic Press, New York, 1975, pp. 237 - 272.

[Schank 84] Schank, R. C. The Explanation Game. Technical Report 307, Yale University Department of Computer Science, New Haven, CT, 1984.

[Schank and Abelson 77] Schank, R. C. and Abelson, R. P. *Scripts, Plans, Goals and Understanding.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

[Silver 86] Silver B. Precondition analysis: Learning control information. In R. S. Michalski, J. G. Carbonell and T. M. Mitchell, Ed., *Machine Learning: An Artificial Intelligence Approach, Volume II*, Morgan Kaufmann, Los Altos, CA, 1986, pp. 647 - 670.

[Williams 86] Williams, G. "The weather watchers." *Atlantic 257*, 1986, pp. 69 - 73.

[Winston 72] Winston, P. H. Learning structural descriptions from examples. In P. H. Winston, Ed., *The Psychology of Computer Vision*, McGraw-Hill, New York, 1972, pp. 157 - 209.

[Winston 80] Winston, P. H. "Learning and reasoning by analogy." *Communications of the ACM 23*, 1980, pp. 689 - 702.

Kepler's and Black's discoveries provide additional examples of numeric laws. The third law of planetary motion relates two observable attributes – the mean *distance* d of a planet from the sun and the *period* p of that planet. Kepler's statement of this law was that "the squares of the periods of revolution of the planets are proportional to the cubes of the mean distance to the sun." However, one can also state this law by defining the term $X = d^3/p^2$ and noting that the value of X is constant across all planets.

Black's heat law is more complex, involving two objects with different temperatures that are placed in contact. Over time, the temperature of one object increases and the other decreases until they become equal. The final temperature is a function of the initial temperatures, the masses of the objects, and the particular substances involved. This law points out the need for our second class of terms – intrinsic properties.

2.2 Defining Intrinsic Properties

An *intrinsic property* is some term that, for a given object or class of objects, has a constant value over time. Thus, this value can be associated with the object/class and retrieved whenever that object/class is encountered. For instance, values of the intrinsic property *mass* are associated with specific objects, while values of the property *density* are associated with entire classes of objects (a 'substance'). Our second operator for empirical discovery is responsible for postulating intrinsic properties and inferring their values.

We denote an intrinsic property as $i\_p(O) = n$, where O is an object or object class and n its associated value for the intrinsic property $i\_p(O)$. Unlike numeric term such as PV/T, intrinsic properties cannot be directly defined in terms of observable attributes. Instead, they require some assumptions about the form of the law involved and the solution of simultaneous equations. However, once an intrinsic property has been defined and its values have been computed, it can be used in the same way as an observable attribute.

After the famous bathtub incident, Archimedes formulated the principle of displacement: the volume of an body immersed in fluid equals the volume of the liquid it displaces. Using this principle, Archimedes was able to measure the volume of an irregular object, and thus to determine its density and composition. This volumetric attribute can be viewed as an intrinsic property for which different irregular objects having different values. Once these values have been determined, they can be used to distinguish different objects from one another.

Mass is another intrinsic property that occurs in several quantitative laws, including conservation of momentum. For the collision of two objects, this law can be stated as:

$$m_1v_1 + m_2v_2 = m_1v_1' + m_2v_2'$$

where m_1 and m_2 are the masses of the two objects, v_1 and v_2 are the velocities before impact, and v_1' and v_2' are the velocities after impact. Given the form of this law and the ability to measure the velocities, we can determine the relative masses of the colliding objects. This involves solving simultaneous equations for the unknown masses, and this in turn requires enough equations to identify their values. If one wants to determine the masses of five different objects, then exactly five observed collisions are needed. Once the mass of an object has been identified, this value can be used in other experiments to discover still other laws.

2.3 Forming Composite Objects

The above operators focus on attributes, and such attributes must always be associated with a single object. However, the conservation of momentum law just described involves a constant relation *between* objects. One way to represent such relations involves defining a new *composite* object, and stating the law in terms of this composite's attributes. Given two or more objects O_1 and O_2, one can define a composite object O_c which has O_1 and O_2 as its components. We express this as:

$$O_c \equiv O_1 \,\&\, O_2$$

Such a composite object can be handled in the same way as an observable object, provided one can determine the values of its attributes. Many of these can be computed directly from the attributes of its component objects. For example, the mass of a composite object is simply the sum of the component masses, while the density involves a weighted average of the component densities.

In summary, composite objects are useful in stating empirical laws which relate some set of objects rather than describing a single object. Our third operator for empirical discovery is responsible for defining such composites. Such an action seems especially useful when a conservation law is involved. Let us consider the momentum example in more detail, in order to clarify the role of this operator and its interaction with the other operators.

The basic experimental situation involves two objects O_1 and O_2 that collide with each other. Based on the initial velocity v and the final velocity v' for each object, our second operator can define the intrinsic properties m (the mass of each object) and infer its value. Based on this property and the velocities, our first operator can define the numeric attributes $P = mv$ (initial momentum) and $Q = mv'$ (final momentum). No simple regularities arise from looking at these attributes for isolated objects. However, if one defines the composite object $O_c \equiv O_1 \,\&\, O_2$, and if one assumes that the momentum of O_c is the sum of its components' momenta, then the simple law $P_c/Q_c = 1$ emerges. This shows some of the representational power one can achieve by defining composite objects.

Now let us consider another example in which there is even more interaction between intrinsic properties and composite objects. If the surface of one body slides over the surface of another, the two bodies exert a frictional force on each other. The quantity of friction depends on the composition of the two objects and on the force pressing the bodies together, but is independent of the area of contact and the speed. This relationship can be expressed as

$$F_f = \mu F_n$$

where F_f is the frictional force, F_n is the normal force pressing the two objects together and μ is the friction coefficient.

The coefficient μ in the friction law can be viewed as an intrinsic property, but unlike most such properties, its values are a function of *both* substances. Thus, the friction coefficient for steel on steel is different than for aluminum on steel, and the best one can do is to store values with each pair of substances. Given this situation, it seems natural to define composite objects* such as steel-steel and aluminum-steel and to associate each intrinsic value with one such composite. In this way, we can retain the assumption that intrinsic values are associated with single objects, and leave the responsibility for creating such objects with our third operator.

2.4 Defining Classes of Objects

Just as one can define composite objects, one can also define new *classes* of objects. Thus, one might decide that objects O_1, O_3, and O_7 have similar properties and belong to the same basic type, leading one to define a new group O_g with these three objects as members.** We will denote this new group as $O_g = \{O_1, O_3, O_7\}$. New terms of this form are quite useful in stating qualitative laws such as occurred in the early days of chemistry and biology. Furthermore, such groups can be modified incrementally; if one later encounters object O_{10} that is similar to existing members of the class O_g, then one may add O_{10} to the class. The process of defining classes can also be applied recursively to form a taxonomy or classification hierarchy. For instance, having defined the object classes O_g and O_h, one might group these together to define a higher level class O_m.

Such taxonomies aid the discovery of qualitative laws at different levels of abstraction. For example, early biologists spent much of their time defining different species, classes of species, and so forth. Similarly, the early chemists devoted considerable effort to defining classes such as alkalis, acids, and salts. In each case, these classes were defined not only by their members, but also by the features held in common by those members. These defining features can be viewed as qualitative empirical laws. Michalski (1980) has used the phrase *conceptual clustering* to refer to this task of formulating taxonomies and determining their associated descriptions.

Just as class formation can help in discovering empirical laws, so can the discovery of qualitative laws suggest new classes. For instance, Mendel experimented with self-fertilized peas and found that some yellow peas produced only yellow offspring, other yellow peas produced both yellow and green offspring, and green peas consistently had green offspring. Based on these observations, he defined the classes of *hybrids* and *purebreds* and formulated the laws of genetic segregation as follows:

* Actually, these are object *classes* rather than individual objects. Just as one can associate intrinsic values with classes of objects as well as specific objects, so can one form composites with object classes.

** Note that the initial objects here are linked to the new object-class by an *instance-of* or *subset-of* relation. This contrasts with the *part-of* relations that holds between composite objects and their components.

$$\forall\, x \in \textit{purebreds}\;\; \textit{parent-of}(x,y) \Rightarrow y \in \textit{purebreds}$$

$$\forall\, x \in \textit{hybrids}\;\; \textit{parent-of}(x,y) \Rightarrow y \in \textit{hybrids} \vee y \in \textit{purebreds}$$

The first of these laws can be paraphrased 'All purebreds produce offspring which are purebreds.' The second empirical generalization can be restated 'All hybrids produce some offspring which are hybrids and some which are purebreds.' The two classes, together with the laws summarizing their behavior, formed the basis for the genetic theory.

As we have mentioned, classes may change their membership over time, and the details of this process may prove interesting. Early chemists first defined the classes of *acids*, *alkalis*, and *salts* in terms of their taste. However, they soon discovered that acids reacted with alkalis to form salts, and this empirical law gradually became a central feature of all three classes. Ultimately, substances that did not taste sour were included as acids because they reacts with known alkalis to form salts. This shift also led to the more abstract class of *bases*; this included the subclasses of alkalis and metals, both of which reacted with acids.

2.5 Defining Composite Relations

Objects can be described by their attributes, but they can also be described through their *relations* to other objects, and this suggests a fifth type of defined term. Given a set of primitive relations between objects, one can define new composite relations. This is similar to the process of defining composite objects, except that one must handle the arguments of these relations.* For example, one can combine the relations *brother(X,Y)* and *spouse(X,Y)* to define the composite relation *brother-in-law(X,Z)*. This can be stated formally as:

$$\textit{brother-in-law}(X,Z) \Leftarrow \textit{brother}(X,Y)\ \&\ \textit{spouse}(Y,Z)$$

this means that X is the brother-in-law of Z if X is the brother of Y and Y is the spouse of Z. By composing an existing relation (such as *parent-of(X,Y)*) and a qualitative attribute (such as *color*), one can also define more specific relation (such as *parent-of-green-child(X,Y)*.

Similarly, one might define the inverse of an existing relation.

The definition of composite relations can be viewed as one form of *chunking*. Although the existing machine learning work on chunking (Neves & Anderson, 1981; Laird, Rosenbloom, & Newell, 1984) has focused on procedural knowledge, chunks can also be perceptually-oriented. In skill acquisition, chunking methods have been used to improve the problem solving process. In scientific discovery, the goal is instead to describe the behavior of objects and classes over time.

* Forming composite relations is also similar to defining numeric attributes, but the latter take at most one object as their argument, while relations can take an arbitrary number. Also, the latter take on only numeric values, while relations describe qualitative links between objects.

Many mathematical concepts can be viewed as relations defined in terms of simpler relations. As Lenat (1977) has shown, one can define multiplication in terms of the addition concept, and one can in turn use multiplication to define the concept *divisors-of*. This term can then be used in the definition of prime numbers, which are simply those natural numbers having only two divisors (themselves and one). Other concepts from number theory can be constructed along the same lines.

In a similar fashion, one can imagine Mendel defining the two restricted versions of the parent relation:

$$\begin{aligned} \textit{parent-of-green-child}(X,Y) &\Leftarrow \textit{parent}(X,Y) \; \& \; \textit{color}(Y,\textit{green}) \\ \textit{parent-of-yellow-child}(X,Y) &\Leftarrow \textit{parent}(X,Y) \; \& \; \textit{color}(Y,\textit{yellow}) \end{aligned}$$

Given these higher-level relations, one can more easily define the classes of purebred and hybrid peas. Purebreds consist of those peas satisfying only one of these relations, while the hybrid class contains those peas satisfying both relations.

2.6 Defining Classes of Relations

If one can define classes of objects, then one can define classes of *relations* as well, and our sixth operator is responsible for this process. Relational classes prove useful in that they can take the place of specific relations in the statement of qualitative laws. For instance, the electrical, magnetic, gravitational, and nuclear forces all differ in their details, but they have much in common as well. As a result, it makes sense to consider them as members of a more abstract force relation. Similarly, both the phlogiston and oxygen theorists held that combustion and rusting were instances of a related process, even though their superficial effects were different. Such relational classes let one state more general laws and make more predictions than a number of specific relations.

Let us consider an example of how this sixth operator might be used. Suppose that one does not yet have a general notion of *reactions*, but knows that when HCl and NaOH are combined, both substances disappear and a new substance NaCl appears (along with some water). Now suppose one combines HNO_3 and KOH, finding that a new substance KNO_3 appears (along with some water). After observing these two experiments, one might define the class of relations *alkali-combines-with-acid-to-form-salt* with these two specific relations as members. One can use this abstract relation in qualitative laws that describe object classes. Moreover, these laws can be used as 'data' in suggesting even more abstract relations, such as the general class of reactions.

The process of defining relational classes is the least well-explored of the operators we have described, and to our knowledge, none of the existing AI discovery systems have used this operator. As a result, it will not come into play during our review in the following section. However, we believe the process of defining relational classes is just as central to constructing a truly integrated discovery system as the other five operators we have discussed.

2.7 An Ordering on the Operators

In the previous sections, we described six operators for defining new terms which form a problem space for empirical discovery. Table 1 lists these operators and the formal notation we have introduced for each. For any reasonable domain, the search space which these operators define is extremely large. Therefore, a robust discovery system will require some *heuristics* to determine the best operator to apply in a given situation.

Table 1: Operators and notation

OPERATOR	*NOTATION*
numeric term	$X = f(a_1, a_3, a_5)$
intrinsic property	$i\_p(O) = n$
composite object	$O_c \equiv O_1 \& O_4$
class of objects	$C_2 = \{O_1,\ O_3,\ O_7\}$
composite relation	$R(O_1, O_2) \Leftarrow R_1(O_1, O_2) \& R_2(O_1, O_2)$
class of relation	$R_c(O_1, O_2) = \{R_1(O_1, O_2),\ R_2(O_1, O_2),\ R_7(O_1, O_2)\}$

As we will see in the following section, existing AI discovery systems address only a subset of this problem space and use at most two of the operators. As a result, the problem of search control is not as serious for these systems.* A more complete response to this problem must take the form of an implemented discovery system which uses all of the operators, thus addressing the entire problem space and forcing a principled answer to search control. Yet a look at the history of science reveals an initial plausible ordering on the operators. Let us review the evolution of chemistry with this goal in mind.

Early chemists were concerned with the classification of chemical substances and with qualitative relations between these substances. This seems natural, since one must decide on a basic set of classes and relations before considering quantitative laws. They formed object classes such as *acids* and *alkalis*, originally defined in terms of simple qualitative attributes but eventually incorporating relational laws. They formed composite relations such as *acid-reacts-with-alkali*, and they also formed abstract classes of such relations. One of the early chemical controversies revolved around whether reactions and

* Actually, Lenat's (1977) AM has an agenda mechanism which lets the system select among tasks. Even though AM uses only two of our operators (defining composite relations and defining classes of objects), this agenda mechanism has the flavor of an integrated system.

mixtures involved two different processes; this can be viewed as a debate about the appropriate classes of relations. Thus, three of our operators – forming object classes, defining composite relations, and forming relational classes – are employed early in the empirical discovery process.

At the end of the 18th century, chemists shifted their attention from qualitative laws to quantitative aspects of chemical reactions. They stopped focusing on symbolic attributes such as color and taste,* and turned to numeric attributes such as volume and weight. This paradigm shift led directly to principles such as the conservation of mass, Proust's law of constant proportions, Dalton's law of simple proportions, and Gay-Lussac's law of combining volumes. These numeric laws related the masses and volumes of the substances involved in reactions, and all were discovered during the late 1700's and early 1800's.

Upon closer examination, we find that the remaining three operators have a central role to play in these quantitative discoveries. For instance, suppose we observe the weight W_E of an element entering a reaction and the weight W_C of the compound that results. From these two terms, one can define the ratio W_E/W_C, and this numeric term has a constant value for any pair of substances. This is one version of Proust's law of constant proportions. Given such a constant value, it make sense to define an intrinsic property and to associate it with the objects involved for future use. However, the value is conditional on both the element and the resulting compound, so that we must first define a composite object and associate the intrinsic value with it. Similar interactions between these three operators occur for Dalton's and Gay-Lussac's laws, and the operator for defining composite objects also proves useful for stating conservation of mass.

To summarize, operators which promote qualitative discoveries (defining classes of objects, composite relations, and relational classes) generally precede operators which promote quantitative discoveries (defining numeric terms, intrinsic properties, and composite objects). However, the ordering on our operators is not as simple as we have suggested. Ultimately, these quantitative discoveries led to higher level 'data' which chemists used formulate higher level classes. In particular, estimates of the intrinsic property atomic weight** led Mendeleev to propose his periodic table, which classified elements using two complementary taxonomies (corresponding to the rows and columns of the table). Hence, qualitative discoveries lay the foundation for quantitative discoveries, but the latter can in turn lead to still higher level qualitative laws.

* It is important to note that qualitative information was not abandoned when chemistry entered its quantitative stage. Qualitative features were still used to identify substances, and such identification was absolutely necessary to successful quantitative studies. However, such identification had become trivial at this point, and the major efforts of chemists were devoted to numeric aspects.

** Actually, qualitative features also played an important role in Mendeleev's discovery, but atomic weight was a central component.

3. Previous Research on Machine Discovery

Now that we have presented a problem space for empirical discovery, let us review some earlier research in this light. Below we review five existing discovery systems. In each case, we begin with an overview of the system. We then consider which of the operators that system employs to discover empirical laws, and examine the conditions under which it applies those operators. We will find that the existing systems search only a small part of the overall space we have defined, never using more than two of the six operators.

3.1 AM

Lenat (1977, 1978, 1982) carried out some of the earliest and best-known research on machine discovery, so it seems appropriate to begin our review by examining his AM system. The program begins with a set of some 125 concepts from elementary mathematics, such as 'set', 'ordered pairs', and 'equality'. Using these as its base, AM defines new concepts in terms of existing ones, arriving at familiar mathematical concepts such as 'natural numbers', 'addition', 'multiplication', and 'prime numbers'. The system also generates hypotheses that relate these concepts to each other, including the unique factorization theorem and Goldbach's conjecture.

AM represents concepts using frame-like structures, each having facets such as *name*, *definition*, and *examples*. The system uses some 250 heuristics (stated as condition/action rules) to guide its search through the space of concepts. These heuristics fall into three general categories – for generating new concepts, for filling in facets of existing concepts, and for determining which task on the agenda to perform next.

Lenat's system incorporates two of our proposed operators – defining composite relations and defining classes of objects. For instance, AM defines the relation of 'addition' in terms of more basic set relations, and then proceeds to define 'multiplication' as repeated addition. The system defines object classes in a model-driven way, generating a new class definition and then running experiments to determine which objects are members of that class. Thus, it defines 'even numbers' to be those 'natural numbers' that can be divided by 2, and then finds that 2, 4, 6, etc. are instances of this class.

Since AM searches a large space of relations and classes, it must restrict its attention to interesting concepts. The system uses several heuristics to this end. One of the most powerful of these rules states that if a relation has been defined in multiple ways, then it is very interesting. For example, AM's searches lead it to define multiplication in four different ways, and this in turn cause the system to devote considerable attention to this concept. Another heuristic focuses AM's processing on object classes which have neither too many nor too few elements. Thus, the system finds the class of primes quite interesting, since there are many examples of this concept, but not too many. In contrast, AM finds the class of even primes to be uninteresting, since it has only one member.

Now let us examine how AM uses these two operators to discover the concept of prime numbers. As we have mentioned, the system finds four alternative definitions for 'multiplication'. This results in a high interest value for the relation, leading AM to spend considerable time examining the concept. One of the system's many heuristics suggests defining the inverse of an interesting relation. AM applies this rule to the current concept, giving

$$\textit{divisors-of}(X,Y) \Leftarrow \textit{multiplication}(X,Y)^{-1}$$

The new relation 'divisors-of' is interesting by its association with multiplication, and AM now invokes another heuristic that suggests looking at extreme cases of interesting concepts. This leads to a number of new objects classes – numbers with zero divisors, with one divisor, with two divisors (the class of primes), and with three divisors. The first two classes turn out to have very few examples, and AM abandons them as a result. However, the system finds that there are few (but not too few) examples of numbers with two divisors and three divisors. Thus, both of these classes are considered interesting enough for further processing.

Upon closer inspection, AM finds a number of relations between these concepts. For instance, numbers with three divisors appear always to be the square of some prime number (a number with two divisors). In addition, the system also finds that every natural number can be factored into a unique set of prime numbers; this is the unique factorization theorem. It also arrives at Goldbach's conjecture that every even number is the sum of two primes. Thus, even though AM spends most of its effort in defining new object classes and relations, it also has the ability to formulate qualitative laws based on these concepts.

AM's search covers only part of the problem space we have defined, but it nevertheless has much of the flavor of an integrated discovery system. The program generates new concepts incrementally, and it designs and carries out its own experiments. It uses these experiments both to uncover qualitative relations and to test hypotheses once they have been formulated. Moreover, AM's agenda mechanism provides a sophisticated strategy for focusing attention and allocating effort. Given this sophistication, it seems surprising that more of our operators did not emerge, but this may be a function of the mathematical domain for which AM was designed.

3.2 BACON

Langley's BACON was another early machine discovery system, though it was actually a series of systems that gradually evolved over the years (Langley, 1978, 1981; Langley, Bradshaw, & Simon, 1983). The emphasis of this work was on general, weak methods for discovering quantitative empirical laws. Given a set of numeric independent and dependent terms, BACON carries out simple 'experiments' to gather data and then searches for one or more empirical laws which summarize those data. The system has discovered a variety of laws from the history of physics and chemistry, including the ideal gas law, Ohm's law for electric circuits, Snell's law of refraction, and Black's heat law. Each of these laws is represented as simple constancies linear relations, and this is where our operators come into play. In order to state complex laws in such a simple format, the system must define terms that make this possible.

To this end, BACON uses two of our operators – defining numeric terms and postulating intrinsic properties. The system's top-level goal is to find some numeric term which has a constant value for the given data, or which is involved in a simple linear relationship. In looking for such terms, BACON carries out a depth-first search through the space of possible terms, with backtracking occurring when necessary. The program limits itself to two types of numeric terms - ratios and products - but these can be applied recursively to define more complex terms involving exponentiation.

Two main heuristics guide the search through the space of numeric terms. One of these rules notes when the values of two terms increase together; in this case, BACON defines the ratio of these terms (unless they are linearly related). Another heuristic notes when the values of one term increases as those of another decrease; in this case, the system defines the product of the two terms. Two final rules note constant values and linear relations; these do not create new terms, but instead formulate empirical laws that incorporate the terms.

For example, given the mean distance d for each solar planet along with its period p, BACON's heuristics note that the values of these terms increase together. This leads the system to define the ratio term $X = d/p$. Upon computing the values of X, BACON notes that these values increase as those of d decrease, and this causes the program to define $Y = dX = d^2/p$. When the values of Y are computed, they are found to increase as those of X decrease, leading to the product $XY = d^3/p^2$. The values of this term are nearly constant across the planets, so BACON formulates a general law that summarizes the original data. The system also includes methods for recursing to higher levels of description in order to find laws involving multiple independent terms, but we do not have the space to discuss them here.

The need for intrinsic properties arises when BACON encounters independent terms with nominal (symbolic) values. Since the system cannot discover a numeric law from symbolic data, it is forced to 'invent' a new numeric term. The values of the intrinsic property are based on the values of the current dependent term. Thus, BACON finds a linear relation between this dependent term and the intrinsic property as soon as the latter is defined, but this relation is tautological. The system can take advantage of the new term to formulate empirically meaningful laws only when its values are used in some different context.

Let us consider an example of intrinsic properties from 18th century chemistry. When Proust began to study the quantitative aspects of reactions, he discovered that a given element always contributes the same percentage to the weight of the resulting compound. Table 2 presents some idealized data which obey Proust's law of constant proportions. For each reaction, the table lists the contributing element, the resulting compound, the weight of the element W_E, and the weight of the compound W_C.

Given these data, BACON first detects that the weight of the element increases with the weight of the compound. This leads the system to define the numeric term $X = W_E/W_C$, which has a constant value for a given element-compound pair. This ratio has a different value for different pairs of substances, but since the element and compound terms take on symbolic values, BACON cannot immediately formulate any further numeric laws. Its response is to define the intrinsic property i_p(Element, Compound) = W_E/W_C and to associate the values of this term (which are based on those of the ratio W_E/W_C) with each particular element/compound pair.* This intrinsic property corresponds to the constant weight ration discovered by Proust.

Table 2: Discovering the law of constant proportions

Element	Compound	W_E	W_C	W_E/W_C
Hydrogen	Water	10.0	90.00	0.1111
Hydrogen	Water	20.0	180.00	0.1111
Hydrogen	Water	30.0	270.00	0.1111
Hydrogen	Ammonia	10.0	56.79	0.1761
Hydrogen	Ammonia	20.0	113.58	0.1761
Hydrogen	Ammonia	30.0	170.37	0.1761

In summary, BACON relies on two of our operators – defining numeric terms and postulating intrinsic properties – and combines these operators in an effective manner. However, the system clearly searches only part of the problem space we have defined, particularly ignoring the importance of qualitative laws and the operators which support their discovery. We would certainly not want to abandon the insights of BACON in future discovery systems, but these insights are certainly incomplete.

3.3 ABACUS

Unlike BACON, which discovers only quantitative relations, the ABACUS system (Falkenhainer 1985, Falkenhainer & Michalski, 1986) combines methods for quantitative and qualitative discovery. ABACUS accepts data in a similar form to those processed by BACON, though it does not require that terms be labeled as independent and dependent. From these data, the system generates numeric laws with qualitative preconditions. As a result, ABACUS can discover multiple laws which hold for different subsets of the data. For example, if a data set contains both liquid and gaseous substances along with their respective pressure, temperature, and volume, the program discovers the following relations:

IF substance = gas	THEN PV/T = *constant*
IF substance = liquid	THEN no relation found

The first of these is equivalent to the ideal gas law, with the condition that the substance be a gas stated explicitly; this version is a more cautious form that found by BACON. The second statement reveals that no analogous law holds for liquids.** Falkenhainer and

* This is not the best example of an intrinsic property, since it does show how such properties can contribute to non-tautological laws. However, it does convey the basic idea.

** This is actually a poor example to distinguish BACON from ABACUS, since the former could actually arrive at similar laws using intrinsic properties if it were given *substance* as a nominal independent attribute. However, ABACUS can also arrive at conditional laws for cases where intrinsic properties cannot be used.

Michalski (1986) present a number of examples of useful preconditions on scientific laws.

ABACUS uses one of our proposed operators, defining numeric terms, to discover quantitative relations between observable attributes. Like BACON, the system searches a space of numeric terms, looking for some term that takes on constant value; the difference is that this term need be constant for only *some* of the observations. In the example above, the numeric term $X = PV/T$ was constant for a subset of the data. In addition to products and ratios, ABACUS also defines new terms by taking sums and differences of existing terms.

The discovery system allows irrelevant variables, but these increase the size of the search space considerably and a simple BACON-like search strategy becomes ineffective. In response, ABACUS employs two new algorithms, *proportionality graph search* and *suspension search*. These search methods will converge on constant numeric terms in a reasonably efficient manner, and include the ability to handle a certain degree of noise. We will illustrate proportionality graph search as it applies to the ideal gas law.

Suppose that we extend the original data set for the discovery of the ideal gas law (the temperature T, volume V, and pressure P of a gas) to include the additional variable M. Further suppose that M is proportional to the volume V, even though this relation is irrelevant to the ideal gas law. ABACUS uses the observations to construct a *proportionality graph* like that shown in Figure 1 for the ideal gas data. The nodes of this graph represent observable variables, while a link between two nodes indicates that these two variables are either inversely or directly proportional to each other. The absence of an edge means that two variables are *not* related. In the figure, there is an edge between V and P because these two variables are inversely proportional to each other. There is no edge between the nodes for M and T, since there is no relation between these variables.

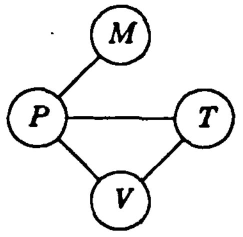


Figure 1: Proportionality Graph for Ideal Gas Law

After ABACUS has constructed this graph, it determines the largest cycle set or biconnected component. For graph in Figure 1, the largest such set is $\{P, V, T\}$. The system then focuses its attention on the members of this set in its attempt to find numeric laws, performing a depth first search with backtracking to find some new term with constant or semi-constant values. A set of heuristics similar to the one used in BACON aids this search. If the largest cycle fails to exhibit such a term, ABACUS defines a new term using variables M and P, includes this term into the set, and continues the search. Falkenhainer and Michalski argue that irrelevant variables are likely to be excluded from such cycles, so that this search will find the desired numeric term more efficiently than a simple depth first search.

Using this search method, ABACUS quickly converges on the constant numeric term $X = PV/T$, despite the presence of the irrelevant variable M. However, the authors concede that proportionality graph search encounters difficulty when complex terms (such as sums of products) are involved. If ABACUS cannot find a useful numeric term using this method, then it resorts to a second search algorithm – suspension search. This process resembles beam search but allows backtracking through the space of terms. Using this approach, the system can discover more complex laws such as conservation of momentum.

ABACUS does not explicitly use our second operator, postulating intrinsic properties, in its search for empirical laws. However, the system's use of logical preconditions leads to effects very similar to intrinsic properties. Consider again the data in Table 1, which led BACON to define the intrinsic property of combining weights and thus to Proust's law of constant proportions. Given the same data, ABACUS would note a relation between the weight of the element W_E and the the weight of the compound W_C and thus define the ratio $X = W_E/W_C$. The system would then notice that this term has semi-constant values, and would set about determining the conditions under which each value occurred. This would produce the following pair of laws:

ClassA	IF	[compound = Hydrogen]
	THEN	$W_E/W_C = 0.1111$
ClassB	IF	[compound = Ammonia]
	THEN	$W_E/W_C = 0.1761$

These state that different values of W_E/W_C are associated with different compounds.* In some sense, these associations are equivalent to those stored by BACON when it postulates intrinsic properties and infers their values. However, there are two important differences. On the one hand, BACON has the ability to retrieve its intrinsic values at some later time and incorporate them into other laws. On the other, BACON can only form intrinsic properties when the nominal variables are under experimental control, while ABACUS can form conditional expressions from observational data.

As we have seen, ABACUS combines methods for qualitative and quantitative discovery, and in this sense it approaches the type of integrated discovery system that is our ultimate goal. However, there are two quite different notions of the term 'qualitative'. Although ABACUS finds qualitative conditions on numeric laws, it does *not* discover laws involving qualitative relations such as those found by the early chemists. The system does not define classes of objects (even though its law-finding methods provide support for this activity), nor does it define composite relations or classes of such relations. Thus, like the other systems so far reviewed, ABACUS searches only a portion of the problem space that we have defined.

We should mention one further point that involves both ABACUS and BACON. As we explained earlier, the operator for defining composite objects can prove quite useful in

* If the data had included different elements as well, ABACUS would have included these in the conditions it discovered.

stating laws such as conservation of momentum. Yet both ABACUS and BACON discover these laws without using this operator. The reason for this apparent inconsistency is that both systems ignore the distinction between objects and their attributes. Rather, they represent data as a conjunction of attribute-values and make no effort to associate attributes with particular objects. In the momentum case, this leads the system to view the given data – the momenta and velocities of the two colliding objects – as belonging to one 'object' rather than two separate objects. Some versions of BACON (Langley, Bradshaw, & Simon, 1982) used subscripts to aid in the search for conservation laws, but this was a weak attempt at best. We believe that future discovery systems would do well to clearly distinguish between objects and their attributes, and to form composite objects when considering a conservation law.

3.4 GLAUBER

As we have already mentioned, much of the effort in an emerging scientific discipline is devoted to classifying objects and to formulating qualitative laws. Langley, Zytkow, Simon, and Bradshaw's GLAUBER (1986) addresses both of these tasks. This system accepts as input a set of qualitative facts, such as taste(HCl, sour) and reacts({HCl NaOH} {NaCl}). GLAUBER transforms these facts into qualitative laws in which specific objects have been replaced by more abstract classes, such as 'acids' and 'alkalis'. These laws also include universal or existential quantifiers that specify the generality of the law.

GLAUBER uses only one of the operators we have described in its formulation of qualitative laws – defining classes of objects. Unlike AM, which first constructs an intensional definition for some class and then generates examples, GLAUBER observes objects in the environment and classifies them based on common features and relations. For example, if a number of objects have the same taste (say sour), the system may define a new class (acids) with these objects as members. GLAUBER then generates a qualitative law which has the same form as the original facts, but in which the class name has replaced the specific objects. Such a law is guaranteed to hold for all members of the class, and so can be universally quantified. However, GLAUBER also substitutes the class for its members in other facts, and in these cases the system must empirically determine whether a universal or existential quantifier is appropriate.

Let us consider GLAUBER's discovery of the concepts of acids, alkalis, and salts. Although the 17th century chemists did not focus on quantitative data, they had considerable qualitative knowledge of substances. This included information about the tastes of various substances, as well as the reactions in which they took part. For example, they knew that HCl had a sour taste and that this substance reacted with NaOH to form the new substance NaCl. These facts and others led the early chemists to group substances like HCl, NaOH, and NaCL into the classes of acids, alkalis, and salts.

Table 3: States generated by GLAUBER in the discovery of acids, alkalis and salts

(a) Initial State

$reacts(\{HCl\ NaOH\},\{NaCl\})$	$taste(NaNO_3,\ salty)$
$reacts(\{HCl\ KOH\},\{KCl\})$	$taste(KNO_3,\ salty)$
$reacts(\{HNO_3\ NaOH\},\{NaNO_3\})$	$taste(NaCl,\ salty)$
$reacts(\{HNO_3\ KOH\},\{KNO_3\})$	$taste(KCl,\ salty)$
$taste(HCl,\ sour)$	$taste(NaOH,\ bitter)$
$taste(HNO_3,\ sour)$	$taste(KOH,\ bitter)$

(b) Intermediate State

$SALTS = \{NaCl, KCl, NaNO_3, KNO_3\}$	
$\exists\, x \in SALTS\ \ni\ reacts(\{HCl\ NaOH\},\{x\})$	$taste(HCl,\ sour)$
$\exists\, x \in SALTS\ \ni\ reacts(\{HCl\ KOH\},\{x\})$	$taste(HNO_3,\ sour)$
$\exists\, x \in SALTS\ \ni\ reacts(\{HNO_3\ NaOH\},\{x\})$	$taste(NaOH,\ bitter)$
$\exists\, x \in SALTS\ \ni\ reacts(\{HNO_3\ KOH\},\{x\})$	$taste(KOH,\ bitter)$
$\forall\, x \in SALTS\ \ taste(x, salty)$	

(c) Final State

$SALTS = \{NaCl, KCl, NaNO_3, KNO_3\}$	$\forall\, x \in SALTS\ taste(x, salty)$
$ACIDS = \{HCl, HNO_3\}$	$\forall\, x \in ACIDS\ taste(x, sour)$
$ALKALIS = \{NaOH, KNOH\}$	$\forall\, x \in ALKALIS\ taste(x, bitter)$

$$\forall\, x \in ALKALIS\ \forall\, y \in ACIDS\ \exists\, z \in SALTS\ \ni\ reacts(\{x\ y\},\{z\})$$

Table 3 (a) presents a similar set of facts that were given to GLAUBER. Examining this initial knowledge base, GLAUBER notices that four of the objects (NaCl, KCl, $NaNO_3$, and KNO_3) have a salty taste, and defines a class with these four objects as members. For the sake of clarity, let us call this class 'salts'. Upon defining this class, GLAUBER replaces instances of the class with the name of the class; this substitution occurs in all facts and laws known to the system. Thus, GLAUBER adds the class SALTS = {NaCl, KCl, $NaNO_3$, KNO_3} to memory, along with the tautological law ∀ x ∈ SALTS taste(x,salty). In addition, the program replaces the salts occurring in reactions with the name of this class, giving a number of more abstract reactions. However, since only one instance of each such pattern occurs, GLAUBER decides on an existential quantifier in each case. The resulting knowledge base is shown in Table 3 (b).

At this point, GLAUBER proceeds to define the class ACIDS = {HCl, HNO_3}, based on the observation that both HCl and HNO_3 have sour tastes. This time, after substitution occurs, the system decides that universal quantification is justified for the reaction laws and it proposes two general laws:

$$\forall\, x \in \text{ACIDS}\ \exists\, y \in \text{SALTS}\ \ni\ \text{reacts}(\{x\ \text{NaOH}\},\ \{y\})$$

$$\forall\, x \in \text{ACIDS}\ \exists\, y \in \text{SALTS}\ \ni\ \text{reacts}(\{x\ \text{KOH}\},\ \{y\})$$

However, these new laws have identical forms, leading GLAUBER to define a third class of substances, ALKALIS = {NaOH, KOH}. This results in the general reaction law shown in Table 3 (c), along with another law describing the taste of alkalis. At this point, the system has successfully summarized all of the original data, so it halts with three classes and four qualitative laws.

Jones (1986) has described NGLAUBER, a successor to GLAUBER that improves on many aspects of the initial system. For instance, GLAUBER required all data to be present at the outset, while NGLAUBER processes data incrementally. In addition, Jones' system is able to distinguish between unobserved facts and disconfirming evidence, such as missing and failed reactions. Although the two systems employ the same operator for defining object classes and formulate similar laws, NGLAUBER uses quite different heuristics than its predecessor. The earlier program operated nonincrementally because it relied on frequency information to decide which classes to form. In contrast, NGLAUBER forms whichever classes are suggested by the most recent data it has examined, but has the ability to backtrack if these classes predict disconfirming evidence. This seems a more plausible model of human scientists than does Langley et al.'s system.

Although GLAUBER and NGLAUBER employ only one of the operators that underly empirical discovery, they fill an interesting niche nonetheless. They show that data-driven heuristics can be used to propose useful classes. They also suggest that some classes are best characterized not by independent features, but by relations between the classes themselves. Finally, the systems point out the need for distinguishing between universal and existential quantification in qualitative empirical laws. We believe that all of these features should be kept in mind in designing more complete, integrated discovery systems.

3.5 OPUS

Another important form of empirical discovery is known as *conceptual clustering*. Basically, this is the task of taxonomy formation, with the added constraint that one formulate an intensional description for each class in the resulting conceptual hierarchy. Since Michalski and Stepp (1983) first defined this problem, a number of conceptual clustering systems have been developed and tested. Rather than attempting to review all of these programs in an already lengthy paper, we will focus on Nordhausen's (1986) recent OPUS system, which has a number of features that are interesting from our perspective.

As we have seen, objects can be described not only in terms of independent attributes, but also through their relation to other objects. OPUS uses both kinds of information to formulate new classes and to find qualitative laws describing those classes. OPUS inputs a set of objects described by nominal attributes such as *color* and *size*, along with binary relations between objects, such as *eat* or *parent*. From these data, the system produces a hierarchical classification tree along with a concept description which uniquely identifies each class.

In constructing this taxonomy, OPUS uses two of the operators we have proposed – defining new classes and defining composite relations. The system defines composite relations in terms of existing relations and simple attributes such as color or size. For example, it combines the binary relation *offspring(X, Y)* and the attribute *color(X,c)* to define the composite relation

$$\textit{offspring-color}(X,c) \Leftarrow \textit{offspring}(X,Y) \ \& \ \textit{color}(Y,c)$$

Once OPUS has defined composite relations, it uses them as attributes during the process of defining object classes. For instance, *offspring-color* can be used to distinguish peas which have only yellow offspring and peas which have both green and yellow offspring. OPUS classifies objects using both primitive attributes (such as color) and attributes that have been derived from relations.

OPUS builds its classification tree in a top-down manner. At each branch the system divides objects into mutually exclusive subclasses, with members having some value of an attribute in common. For example, if the attribute 'color' is used to partition objects, OPUS divides the objects into classes with members of the same color. The program then selects that attribute which best divides the current object set according to two criteria. The *simplicity* criterion favors classes with simple descriptions, while the *inter-cluster difference* criterion promotes classes with different properties. If none of the existing attributes can distinguish between the existing set of objects (i.e., if members of all classes have the same value for the given attributes), then OPUS defines new attributes and uses these to define new classes. This process is recursive, so that defined attributes can be used as the basis for more complex attributes.

Now that we have described OPUS in the abstract, let us examine its use of the two operators in rediscovering the classes of hybrids and purebreds from the early days of genetics. In this domain, OPUS is provided with information about the color of various peas (green or yellow), along with the parent-child relations between different peas. For example, pea A might be described as $\textit{color}(A, \textit{green})$ and $\textit{parent}(A, B)$. At the outset, OPUS uses the primitive attribute color to define the classes of yellow peas and green peas. But because no distinctions can be made on the basis of existing attributes, the system defines two composite relations for this purpose: *offspring-color(X,c)* and *parent-color(X,c)*.

Both relations can then be used as attributes to refine the existing classes. In this case, the attribute *offspring-color* does a better job of partitioning the objects, so OPUS selects this term to extend the classification tree. As a result, the system refines the class of yellow peas into two subclasses – those which produces only yellow offspring and those which produce both yellow and green offspring. At this point, OPUS has not only formulated the classes of hybrids and purebreds; it has also described these classes using concepts very similar to the ones proposed by Mendel.

Elements of the class of purebreds have purebred offspring.
Elements of the class of hybrids have purebred and hybrid offspring.

OPUS continues this process, further refining the purebred class into those with hybrids as parents and those with purebreds as parents. Figure 2 presents the final taxonomy generated by the system; this is very similar to the organization proposed by Mendel in the 1860's.

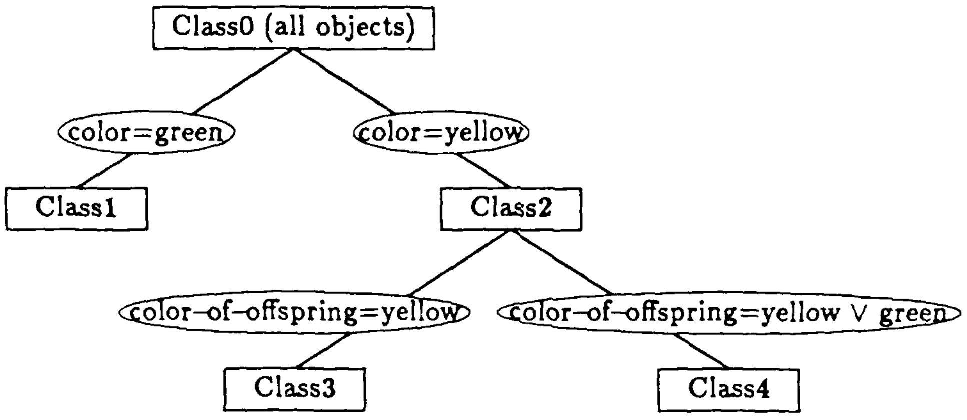


Figure 2: Classification tree equivalent to Mendel's definitions

OPUS is interesting along a number of dimensions relevant to our framework. Like AM, this system defines both object classes and new relational terms. However, it applies these operators in quite different contexts and to quite different ends than did Lenat's early system. Nor is OPUS a traditional conceptual clustering system, since it focuses on relations between objects as well as isolated features of those objects. But the most interesting aspect of the system lies in the interaction between the two operators. OPUS defines composite relations in order to support the creation of new object classes, just as BACON postulates intrinsic properties in order to allow the creation of useful numeric terms. This is precisely the type of interaction we would hope for in an integrated system, in which each of the six operators feed off the results of the others to create powerful synergies that aid the discovery process.

3.6 Summary

In this section, we reviewed five existing empirical discovery systems in the light of our framework. We summarize the results of this analysis in Table 4. Cells marked with crosses indicate operators that clearly exist within the specified system, while triangles indicate ambiguous cases where the operator is absent, but where the system achieves a similar effect indirectly. The most obvious characteristic of the table is its sparsity; very few of the possible cells are occupied. In fact, none of the systems incorporate more than three of the operators, even with a liberal interpretation.

Table 4: Discovery systems and their operators

System	*numeric term*	*intrinsic property*	*composite object*	*class of objects*	*composite relation*	*class of relations*
AM				×	×	
BACON	×	×	△			
ABACUS	×	△	△			
GLAUBER					×	
OPUS				×	×	

This means that each of these AI discovery systems search only a portion of the problem space of defined terms that we described earlier, and this limits the class of laws that each system can discover. This in turn suggests a natural goal for future research – the design and construction of an integrated discovery system that employs all six operators to search the entire problem space. In the following pages, we describe our plans for such a system.

4. An Integrated Model of Empirical Discovery

Although we believe our framework for empirical discovery has helped to clarify and unify earlier work in the area, it has only limited usefulness. Our ultimate goal is to translate this framework into an integrated system discovery system. Only by following this path can we determine whether our operators are necessary and sufficient for empirical discovery, and identify heuristics to direct the application of these operators in an intelligent fashion. In this section, we detail our plans for an integrated discovery system (IDS) that incorporates all six of the operators we have proposed. However, in order to realistically simulate the discovery process, one needs some environment which is separate from the discovery system, but which that system can inspect and manipulate. We are implementing such an environment for the domains of early physics and chemistry which obeys the major laws of these domains. Below we describe the environment in some detail, before turning to our designs for the discovery system itself.

4.1 Objects and Attributes

The simulated environment contains a set of *objects*, each having a variety of attributes. These attributes are similar to those available to early physicists and chemists, such as volume, color, taste, shape, location, temperature, and mass. Many of these attributes are numeric in nature, but others (like color and taste) are usually viewed as nominal (symbolic). However, we have also chosen to represent these as numeric terms with real values, since we feel this more closely reflects the situation encountered by the early scientists. Thus, the taste of an object involves three sub-attributes – saltiness, sourness, and bitterness – each taking values from zero to one. We use similar sub-attributes to represent the colors of objects.

A few attributes seem genuinely nomi al, at least for our purposes. For instance, the *state* of an object can be *solid*, *liquid*, or *gaseous*. These values represent qualitatively different aspects that one can determine through direct inspection. Similarly, the *shape* of an object takes on the nominal values *box*, *sphere*, *cylinder*, or *irregular*. Although these certainly do not exhaust the possible shapes occurring in the physical world, they provide enough variety to allow interesting behavior.

In addition, primitive objects can be connected to form more complex composite objects.* Thus, one can specify that two or more primitive objects are *parts* of a complex object. These components must move together and are affected together along other dimensions (such as temperature). The environment supports three forms of object composition. *Generic* composition simply specifies that two objects are part of a composite object, but the two other forms specify additional features. Composition by *containment* specifies that one object is contained by another. This is essential if our system is to replicate early chemical discoveries involving gases and liquids. Similarly, two containers may be *connected* by a conduit, allowing the contents to move from one object to the other. These relations let one construct reasonably complex systems of objects. Finally, two objects can *touch* one another; this relation does not define a composite object, but many laws include adjacency as an application condition.

An important aspect of the environment is that it changes over time. Thus, the temperature of object A at one instant may differ from its temperature at the next instant. Some attributes may well have constant values, but this is something the system must discover for itself. In other cases, the system must formulate laws that describe an object's change over time. In addition, new objects may enter the world and existing objects may disappear (as in chemical reactions). The discovery system must be able to summarize these qualitative changes as well as quantitative ones. These possibilities will force us to handle laws and explanations of a quite different nature than those we addressed in previous research.

4.2 Gathering Data and Performing Experiments

The discovery system will observe the world through a set of *sensors*. These are passive in nature, simply letting the system inspect the value of an object along a certain dimension; they correspond to primitive measuring instruments, such as rulers, scales, and thermometers. In general, one sensor exists for each observable attribute. Thus, at any given time, the system can measure the following properties of any given object: mass, temperature, color (lightness, hue, saturation), taste (saltiness, sourness, bitterness),

* We are talking here about the physical combination of objects. The reader should not confuse this with our third operator, which involves the *logical* composition of objects.

location (x and y coordinates), size (radius; length, width, depth), texture, shape, and state.

Some sensors can be applied only to certain objects. For instance, the system can inspect the radius of spherical objects and the length, width, and depth of boxes, and from this one can easily compute their volumes. However, one cannot directly measure the dimensions of irregular objects, and this makes the derivation of volume more difficult. Restrictions also apply to the components of complex objects. The system can measure the color, temperature, and locations of the components independently, but it cannot directly measure these values for the composite object. On the other hand, it can measure the mass of composite objects, but not the mass of their components. The system may be able to infer these values, but this requires intelligent behavior rather than simple sensing.

Most earlier discovery systems were provided with data, but in this environment one must actively gather information. If the system wants to measure the mass of object A during some time cycle, it must explicitly call on its mass sensor with A as the argument. Moreover, the number of such measurements that can be made during a given cycle is limited.* Thus, the system must focus its attention on objects and aspects of those objects that it decides are important.

In addition to sensors, the simulated environment also supports active processes called *effectors*. These let one affect objects directly, including actions such as changing the location of an object, breaking an object into two equal components of the same type, and heating an object. Like sensors, the effectors require an intentional act on the part of the system. These actions also let the system construct composite objects using the composition relations (generic, containment, and connection) described earlier. Thus, one can construct simple experimental configurations by rearranging objects and their relations to each other.

More important, one can run simple experiments by creating initial conditions and then using sensors to observe changes over time. For instance, one might place two objects in contact and heat them both. In some cases, a new object with different features will be created, and the mass of this object will increase over time as the masses of the original objects decreases. In this way, we can simulate simple chemical reactions.

Note that such experiments provide a way of defining new measuring instruments. Thus, one might measure the volume of liquid held in some container, place an irregular object in the container as well, and then measure the resulting volume. Archimedes used a similar strategy to measure the volumes of irregular objects, and this ability provides an interesting range of behaviors that have been largely ignored in work on machine discovery. Another example of a new 'measuring instrument' involves sensing the temperature of an object, heating it at constant rate for some time, and resensing the temperature. Together with the elapsed time, these temperatures let one estimate the specific heat of the object.

Now that we have described the environment in which our discovery system (IDS) will

* We plan to start by allowing 10 sensors to be applied simultaneously, but we may reduce or increase this limit based on our experience.

operate and the primitive actions it has available for interfacing with that environment, let us turn to the system itself. We have divided our discussion into two parts, the first dealing with qualitative discovery and the second handling the formulation of quantitative laws. We will see that all six of our operators are embedded within the design of IDS, and that the system's methods for numeric discovery build naturally upon the qualitative laws it constructs at the outset.

4.3 Inferring Qualitative Schemas from Behavior

Before it can discover numeric relations, our discovery system must first determine the basic types of events that occur in its surroundings. The system will begin by examining individual objects, looking for terms that are constant over time. Most attributes of objects will be constant over time until some effectors are applied. Based on these constancies, the system will generate an initial taxonomy, grouping similar objects together. This activity corresponds to the operator for defining classes of objects. The first such classes will be chemical substances, the members of which have the same color, texture, taste, and density (a defined term), but which have different masses and volumes. More abstract classes such as metals (which are smooth and shiny) and acids (which taste sour) may also be defined, but members of these groups will have fewer features in common.

Once an initial set of classes have been identified in this manner, the system will use them in designing experiments and in generalizing the results of those experiments. This involves applying effectors to members of different groups and observing the results. Let us consider a simple experiment as an example. Suppose one fills container C_1 with liquid L_1 to height H_1 and fills container C_2 with liquid L_2 (of the same class) to height H_2, and then connects these two containers with an open conduit. As time passes, one observes the heights of liquid in each container, noting that one level increases and the other decreases until the two levels are equal, having reached equilibrium.

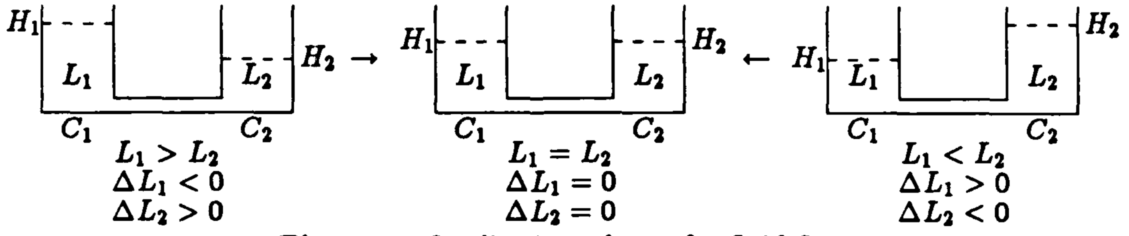


Figure 3: Qualitative schema for fluid flow

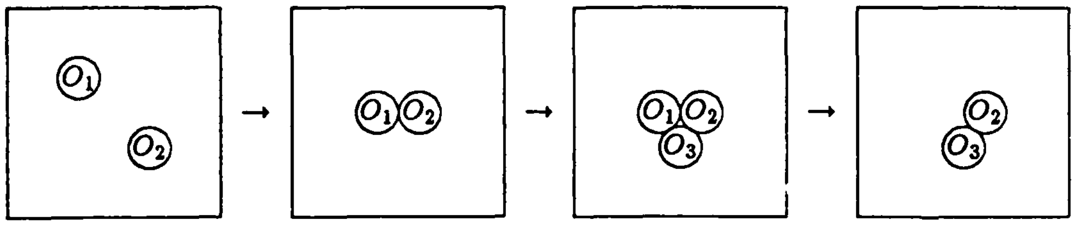
If we focus on the qualitative aspects of this situation, only two classes of states exist. The first class can be described by three relations: $L_1 > L_2$, $\Delta L_1 < 0$, and $\Delta L_2 > 0$. Similarly, the second (equilibrium) class of states can be described by different relations: $L_1 = L_2$, $\Delta L_1 = 0$, and $\Delta L_2 = 0$. These classes can be easily induced from the manner in which the system changes over time. Moreover, one can also infer that the first class of states leads to the second class; in other words, any non-equilibrium situation is gradually transformed into an equilibrium situation. We represent this qualitative schema graphically in Figure 3; the illustration includes the alternative situation, in which one begins with $L_1 < L_2$.

The representation we have used for this qualitative schema is very similar to that proposed by Forbus (1984) in his qualitative process (QP) theory. However, note that we have no model of the *processes* responsible for the transition between states in our schema. Rather than inferring the schema from process knowledge (as Forbus does with his envisionment mechanism), IDS will induce the schema by observing changes in the environment over time. In some sense, our schema represents process knowledge in its own right, but uses a form quite different from that used in QP theory.

Now let us consider a more complex example involving a chemical reaction. Suppose we move two objects O_1 and O_2 into contact with each other, and that a new object O_3 is generated as a result. Moreover, imagine that the masses of O_1 and O_2 decrease over time until O_1 reaches zero (and thus disappears), while the mass of O_3 increases in the meantime. Finally, suppose the reaction ends with the masses of O_2 and O_3 remaining constant over time.

As before, we can represent these changes with a qualitative schema like the one shown in Figure 4. The first box shows the initial class of states during which O_1 and O_2 are being moved closer together. Letting D be the distance between two objects and M be the mass of an object, a number of change relations hold during these states: $\Delta D(O_1, O_2) < 0$. $\Delta M(O_1) = 0$, and $\Delta M(O_2) = 0$. Note that we include terms with constant derivatives, provided these derivatives change elsewhere in the schema. After the two objects have been brought together, the new relation $\Delta D(O_1, O_2) = 0$ replaces $\Delta D(O_1, O_2) < 0$, since the relative positions of the objects are constant.

The transition from the second class of states to the third class introduces the new object O_3. We believe that the creation or destruction of an object is always sufficient justification for establishing state boundaries. Moreover, the qualitative relations have changed again. During this class of states, the distances between objects remain the constant zero, but the masses change: $\Delta M(O_1) < 0$, $\Delta M(O_2) < 0$, and $\Delta M(O_3) > 0$. In the transition to the final state-class, the object O_1 is destroyed, and the masses of O_2 and O_3 remain constant during these states. Taken together, these successive state descriptions form a qualitative description of the events that occur during a simple chemical reaction.



State 1	State 2	State 3	State 4
$D(O_1, O_2) > 0$	$D(O_1, D_2) = 0$	$D(O_1, D_2) = 0$	$M(O_1) = 0$
$\Delta D(O_1, O_2) > 0$	$\Delta D(O_1, O_2) = 0$	$\Delta D(O_1, O_2) = 0$	$\Delta M(O_2) = 0$
$\Delta M(O_1) = 0$	$\Delta M(O_1) = 0$	$\Delta M(O_1) < 0$	$\Delta M(O_3) = 0$
$\Delta M(O_2) = 0$	$\Delta M(O_2) = 0$	$\Delta M(O_2) < 0$	
		$\Delta M(O_3) > 0$	

Figure 4: State description of chemical reaction

Although IDS will form such qualitative schemas on the basis of a single experiment, note that the resulting description is quite general. In fact, one can view the above process as defining a composite relation; this is one of the six operators we discussed earlier.* Thus, IDS might use the name *reacts* to refer to the qualitative schema in Figure 4, and specify a successful instantiation of the schema involving objects O_6, O_7, and O_9 as $reacts(O_6, O_7, O_9)$. Such a representation could be passed directly to a GLAUBER-like subroutine, which would define new classes of objects and formulate qualitative laws.

Of course, one must still carefully select the objects used in the experiments to maximize the likelihood of useful results. However, recall that IDS will have already grouped objects into initial classes based on common features, and it can use these classes to constrain the process of experimentation. For instance, the system might decide to combine members of the class of sour-tasting objects (acids) with each other, but no reaction would occur in these cases and it would give up after a few unsuccessful attempts. However, the system would have more success when combining acids with members of the bitter-tasting class (alkalis). Moreover, the outputs of these reactions (salts) may never have been observed before, giving IDS a new class of objects to use in other experiments.

4.4 Finding Quantitative Laws

Once a qualitative schema has been formulated, it provides the *context* within which numeric laws can be framed. One of BACON's drawbacks was that it failed to specify the situations under which its quantitative laws held, and IDS's qualitative schemas provide a formalism for doing this. In particular, each of the qualitative relations that occur in the schema may be transformed into a quantitative law, which is then attached to that class of states. For instance, in our equilibrium example we found that the level of one liquid decreased as the level of the other decreased. A numeric law might specify the exact rates at which these changes occurred. Another numeric law might state the final level of equilibrium as a function of the initial levels of the liquids.

Thus, IDS would repeat the same 'experiment' with different numeric parameters, instantiating the same qualitative schema in different ways. In the equilibrium example, the system could fill the containers to different initial levels and observe the resulting rates of change and equilibrium states. In the chemical reaction example, it could not use the same objects, since these are transformed during the reaction, but it could use the same classes of objects (such as *ammonia* and *sulfuric acid*). In this case, it would vary the initial masses involved in the reaction and observe the masses remaining afterwards.

We envision IDS using BACON-like heuristics to direct the search for numeric laws. The system would consider the product of two terms if they increase together and consider their ratio if one increases as the other decreases. Our experience with BACON suggests that such heuristics are quite robust even in the presence of significant noise, provided the

* In some sense, the generality of these schemas makes them *classes* of relations. Rather than starting with specific schemas and forming more general ones, we envision IDS as starting with very general relations which share the same qualitative descriptions. The system would then gradually form more specific versions of these schemas that differ in their quantitative features.

laws involve only a few parameters. In addition, once IDS has discovered a numeric law for one object/class or pair of objects/classes, it will predict that the same law will hold for other objects/classes, even though the numeric parameters differ. When this occurs, the system will associate each value with the object or class, storing it as an intrinsic value that may be retrieved in other situations as well. Thus, IDS will include two more of the operators for empirical discovery – defining numeric terms and postulating intrinsic properties.

Let us consider the example involving chemical reactions in more detail. Suppose IDS places an object from the nitrogen class into contact with another object from the oxygen class, and that the object which emerges from the reaction has features of the nitric oxide class. Further, suppose the system runs this same basic experiment with different amounts of nitrogen (say 1.0 gram, 2.0 grams, and 3.0 grams) while holding the amount of oxygen constant at 6.0 grams. Each of these experiments will obey the qualitative schema shown in Figure 4, with object O_3 (nitric oxide) being created and object O_1 (nitrogen) disappearing.

Upon examination, IDS would find varying amounts of oxygen in each case (4.86 grams, 3.72 grams, and 2.58 grams). Comparing these values to the masses of nitrogen used in each case, it would note a linear relation with slope -1.14 and an intercept of 6.0. Varying the initial amount of oxygen causes the intercept to vary, but the slope remains constant at -1.14. This constant term corresponds to the *combining weight* of oxygen with respect to nitrogen when these two chemicals combine to form nitric oxide. Based on this constancy, the system would define an intrinsic property and associate this particular value with the nitrogen-oxygen-nitric oxide triple.* Different intrinsic values for this term would be found for other chemical reactions that obeyed the same qualitative schema.

Taken together, the linear relation and intrinsic property specify a numeric law that describes the quantitative behavior of the schema in Figure 4. This law relates the $M(O_2)$ term occurring in the final class of states to the $M(O_1)$ term occurring in the original state-class. The IDS system would discover similar laws relating the final value for $M(O_1)$ (when this object remains) to the initial value of $M(O_2)$, and relating the final value for $M(O_3)$ to the initial values for $M(O_1)$ and $M(O_2)$.** These laws correspond to Proust's law of constant proportions. We have not considered the changes that occur in volume along with changes in mass, but if IDS focused on this term as well, it would also arrive at Gay-Lussac's law of combining volumes.

Although BACON rediscovered both Proust's and Gay-Lussac's laws, it did so in a much different form than just described. Both its data and its laws were stated in very abstract terms, divorced from any description of the physical situation involved. In the new framework, the data consist of instantiations of the given qualitative schema, and the

* Rather, it would define a composite object with nitrogen, oxygen, and nitric oxide as components, and associate the intrinsic value with this new object. This constitutes another of our six operators.

** In fact, the procedure of combining two objects through a chemical reaction and measuring the slope of the line relating their masses can be viewed as a new, higher level sensor for measuring combining weights. In some sense, the system will have defined a new measuring instrument.

laws relate numeric terms that occur in that schema. In addition to providing a context for numeric laws, such schemas also make possible a new class of relations that BACON did not consider – laws describing rates of change. Since the initial qualitative relations are described in terms of derivatives, it seems natural for the quantitative component of IDS to identify the constants associated with these derivatives, and (if they exist) to store them as intrinsic properties of the objects or classes involved in the reaction. We plan to explore methods for discovering such laws as well, though we have not yet formulated the details.

4.5 Summary

In this section, we outlined our plans for IDS, an integrated discovery system that instantiates the framework we proposed earlier in the paper. The system will interact with a simulated physical world through a set of sensors and effectors, and these will let IDS implement simple experiments and design new measuring instruments. In addition, the environment will change over time, forcing IDS to represent and discover types of laws that earlier machine discovery systems have ignored. The program will focus first on defining useful classes of objects, as well as determining qualitative schemas that describe changes over time. Once these schemas have been established, they will provide the context for discovering numeric laws.

Although our concern here has been with empirical discovery, IDS's schema representation also suggests an approach to theory formation. We have focused on empirical laws that deal with macroscopic events in which one can directly observe objects and changes in those objects. However, much of scientific discovery involves formulating explanations of laws and behavior in terms of structures and events that *cannot* be observed. The caloric theory and the kinetic theory of gases are two well-known examples of such explanations. Basically, we believe that explanatory theories can be formed through a process of analogy with schemas based on macroscopic phenomena. These analogies are cued by similar qualitative changes, and lead one to infer physical structure (such as the coloric fluid) that are not directly observable.

We do not have the space to consider this process in detail, and our ideas on theory formation are still rather vague in any case. But we find it encouraging that the notion of a qualitative schema may prove useful in theory formation as well as during the discovery of empirical laws. This suggests that our design for IDS will prove a fertile one for modeling the process of discovery.

5. Conclusions

Scientific discovery is a complex phenomenon involving many interacting components. Even the process of empirical discovery is sufficiently complex that earlier research on machine discovery has addressed only parts of the overall task. In this paper, we presented a general framework for empirical discovery that we hope will further our understanding of this process. Like much of the work in AI and machine learning, our framework is based upon the notion of a problem space, and we have spent much of the paper describing the operators that define that space. But rather than focusing on operators for law discovery per se, as one might expect, we focused instead on operators for defining new terms. There is ample precedent for this, since the existing machine discovery systems spend more effort in finding useful terms than they do in finding empirical laws.

We proposed six types of terms that prove useful in empirical discovery, each with an associated operator responsible for its definition. We attempted to justify each of these types with examples from the history of science, and we also used historical data to suggest a possible ordering on the operators. We found that all but one of the operators had been used in existing machine discovery systems, but that none of these systems employed more than three of the operators. In other words, previous research on machine discovery has limited itself to small portions of the total problem space. This has been a useful strategy, but we feel the time has come to construct an integrated discovery system that explores the entire space of terms and thus discovers a much wider range of laws.

In fostering this effort, we have constructed a simulated environment with which our integrated system (IDS) will interact. The system will have sensors for measuring directly observable attributes of objects, as well as effectors for running simple experiments. Objects in the environment will change over time, introducing a factor that has been absent from earlier AI work on discovery. Within this framework, IDS will begin by constructing qualitative schemas (composite relations) that summarize changes over time. The system will run experiments to determine which objects obey these schemas, and this in turn will lead to classes of objects and relations.

Once such a qualitative schema is well understood, IDS will attempt to determine the quantitative laws that govern that schema. This will lead the system to define numeric terms, intrinsic properties, and composite objects. Moreover, the schema will provide a context within which such numeric laws can be interpreted; this is quite different from the abstract quantitative relations formulated by BACON and ABACUS. Finally, we have plans to move beyond empirical discovery and into the realm of explanation, using the same representation of events for empirical laws and scientific theories.

We believe this approach will lead to a robust and integrated system for empirical discovery, but our work on this system is still in the planning stages. The most important part of the effort remains; we must translate our ideas into a running program, and we must test this system on a wide range of discovery tasks to ensure its power and generality. However, we believe that our framework for empirical discovery has already proved useful in both clarifying earlier work in the area and in proposing directions for more powerful systems. But the approach we are taking with IDS is not the only instantiation of this framework. We encourage our colleagues to develop other approaches to empirical discovery that explore the same problem space using different methods. Working together, we can achieve both a broader and a deeper understanding of the complex phenomenon called 'discovery'.

Acknowledgements

We would like to thank Randy Jones, Don Rose, and John Gennari for their comments on an early draft of this paper and for discussions that led to the ideas presented therein. This research was supported by Contract N00014-84-K-0345 from the Information Sciences Division, Office of Naval Research.

References

Arons, A. B., *The Development of Concepts of Physics: The Rationalization of Mechanics to the First Theory of Atomic Structure*, Addison–Wesley, Reading, MA, 1965.

Bradshaw, G. L., Langley, P., and Simon, H. A., "Studying Scientific Discovery by Computer Simulation," *Science*, Vol. 222, No. 4267, Dec. 1983.

Dampier, W. C., *A History of Science*, MacMillian, New York, 1943.

Falkenhainer, B. C., "Proportionality Graphs, Units Analysis, and Domain Constraints: Improving the Power and Efficiency of the Scientific Discovery Process," *Proceedings of the Eight IJCAI*, Los Angeles, pp. 552–554, 1985.

Falkenhainer, B. C. and Michalski, R. S., "Integrating Quantitative and Qualitative Discovery: The ABACUS System." To appear in *Machine Learning* 1986.

Forbus, Kenneth D., "Qualitative Process Theory," Technical Report No. 789, AI Lab, MIT, 1984.

————, "Interpreting Observation of Physical Systems," Report No. UIUCDCS–R–86–1248, University of Illinois at Urbana–Champaing, 1986.

Jones, R., "Generating Predictions to Aid the Scientific Discovery Process." To appear in *Proceedings of AAAI-86*, Philadelphia, PA, 1986.

Laird, J. E., Rosenbloom, P.S. and Newell, A., "Towards Chunking as a General Learning Mechanism," *Proceedings of AAAI-84*, Austin, TX, pp. 188–192, 1984.

Langley, P., "BACON.1: A General Discovery System," *Proceedings of the Canadian Society for Computational Society of Intelligence*, Toronto, pp. 173–180, 1978.

————, "Data–Driven Discovery of Physical Laws," *Cognitive Science*, Vol. 5, pp. 31–54, 1981.

Langley, P., Bradshaw, G. L. and Simon, H. A., "BACON.5: The Discovery of Conservation Laws," *Proceedings of the Seventh IJCAI*, Vancover, B.C., pp. 121–126, 1981.

————, "Data–Driven and Expectation Driven Discovery of Empirical Laws," *Proceedings of the Canadian Society for Computational Society of Intelligence*, Saskatoon, Saskatschewan, pp. 137–143, 1982.

Langley, P., Bradshaw, G. L. and Simon, H. A., "Rediscovering Chemistry with the BACON System," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds., Tioga Press, Palo Alto, CA, pp. 307–329, 1983.

Langley, P., Bradshaw, G. L., Zytkow, J. M., and Simon, H. A., "Three Facets of Scientific Discovery," *Proceedings of the Eighth IJCAI*, Karlsruhe, W. Ger., pp. 465–468, 1983.

Langley, P., Simon, H. A., Zytkow, J. M., and Fisher, D. H., "Discovering Qualitative Laws," Technical Report 85-15, Department of Information and Computer Science, University of California, Irvine, 1985.

Langley, P., Zytkow, J., Simon, H. A., and Bradshaw, G. L., "The Search for Regularity: Four Aspects of Scientific Discovery," in *Machine Learning: An Artificial Intelligence Approach, Vol. II*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), Morgan Kaufmann, Los Altos, CA, pp. 425–469, 1986.

Leicester, H. M. and Klickstein, H. S., *A Source Book in Chemistry*, Harvard University Press, Cambridge, MA, 1963.

Lenat, D. B., "Automated Theory Formation in Mathematics," *Proceedings of the Fifth IJCAI*, Cambridge, MA, pp. 833–841, 1977.

————, "The Ubiquity of Discovery," *Artificial Intelligence*, Vol. 9, No. 3, pp. 257–285, 1978.

————, "AM: Discovery as Heuristic Search," in *Knowledge-Based Systems in Artificial Intelligence*, R. Davis and D. B. Lenat, McGraw-Hill, New York, 1982.

Lenat, D. B. and Brown, J. S., "Why AM and EURISKO Appear to Work," *Artificial Intelligence*, Vol. 23, No. 3, pp. 269–294, 1984.

Magie, W. F., *A Source Book in Physics*, Harvard University Press, Cambridge, MA, 1963.

Michalski, R. S., "Knowledge acquisition through conceptual clustering: A theoretical framework and algorithm for partitioning data into conjunctive concepts," *International Journal of Policy Analysis and Information Systems*, Vol. 4, pp. 219–243, 1980.

Michalski, R. S. and Stepp, R. E., "Learning from observation: Conceptual clustering," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds., Tioga Press, Palo Alto, CA, pp. 331–363, 1983.

Neves, D. M. and Anderson, J. R., "Knowledge Compilation Mechanism for the Automatization of Cognitive Skills," in *Cognitive Skills and Their Aquisition*, Lawrence Erlbaum Associates, Hillsdale, N. J., pp. 57–84, 1981.

Nordhausen, B., "Conceptual Clustering Using Relational Information." To appear in *Proceedings of AAAI-86*, Philadelphia, PA, 1986.

Singer, C., *A Short History of Science to the Nineteenth Century*, Clarendon Press, Oxford, 1941.

The AQ15 Inductive Learning System: An Overview and Experiments

by

Ryszard S. Michalski, Igor Mozetic[1], Jiarong Hong[2], Nada Lavrac[1]

July 1986

Intelligent Systems Group
Department of Computer Science
University of Illinois at Urbana-Champaign

This research was supported in part by the National Science Foundation under Grant No. DCR 84-06801, the Office of Naval Research under Grant No. N00014-82-K-0186, the Defense Advanced Research Project Agency under Grant No. N00014-K-85-0878, and by the Slovene Research Council.

[1]On leave from *Jozef Stefan Institute, Ljubljana, Yugoslavia*
[2]On leave from *Harbin Institute of Technology, Harbin The People's Republic of China*

APSTRACT

AQ15 is a multi-purpose inductive learning system that uses logic-based, comprehensible knowledge representation. It is able to incrementally learn attributional disjunctive concepts from data that may contain erroneous or inconsistent examples, and can perform constructive induction. The latter means that the program uses background knowledge to generate new attributes not present in the input data, and, if they pass a relevance test, employs them in the learning process. In an experimental application to three medical domains, the program learned decision rules that performed at the level of accuracy of human experts. A surprising and potentially significant result is the demonstration that by applying the proposed method of rule reduction and flexible matching (TRUNC), one may drastically decrease the complexity of the knowledge base without affecting its performance accuracy.

Keywords: Knowledge Acquisition, Machine Learning, Inductive Inference, Applications (Medicine)

1. INTRODUCTION

It is widely acknowledged that the construction of a knowledge base represents the major bottleneck in the development of any AI system. An important method for overcoming this problem is to employ inductive learning from examples of expert decisions. In this knowledge acquisition paradigm, knowledge engineers do not have to force experts to state their "know how" in a predefined representational formalism. Experts are asked only to provide correct interpretation of existing domain data or to supply examples of their performance. It is known that experts are better at providing good examples and counterexamples of decisions than at formalizing their knowledge in the form of decision rules. Early experiments exploring this paradigm have also shown that decision rules formed by inductive learning may outperform rules provided by human experts [Michalski & Chilausky 80, Quinlan 83].

This paper describes briefly an inductive incremental learning program AQ15 that learns attributional descriptions from examples. As an important aspect of development of learning systems is their evaluation using practical problems, we also present results of applying AQ15 to three medical domains: lymphography, prognosis of breast cancer recurrence, and location of primary tumor. These three domains are characterized by consecutively larger amounts of inconsistent and sparse learning events.

The evaluation was done from the viewpoint of *classification accuracy* of the induced rules on new objects and *complexity* of the rules. Examples of a few hundred patients with known diagnoses were available, along with the assessed classification accuracy of human experts. We randomly selected 70% of examples for rule learning and used the rest for rule testing. For each domain, the experiment was repeated four times. The induced rules reached the classification accuracy of human experts. Performance of experts was measured in two out of three domains (breast cancer and primary tumor) testing four and five experts, respectively. The experiments revealed an interesting phenomenon that by *truncating* rules and applying *flexible* rule matching one may significantly reduce the size of the knowledge base without decreasing its performance accuracy.

2. DESCRIPTION OF AQ15

The program AQ15 is a descendant of the GEM program [Reinke 84] and the AQ1-AQ11 series of inductive learning programs, e.g., [Michalski & Larson 75]. Its ancestors were experimented with in the areas of plant disease diagnosis [Michalski & Chilausky 80, Reinke 84], chess end games [Reinke 84], diagnosis of cardiac arrhythmias [Mozetic 86], and others. This section provides a brief description of AQ15 and its basic features. A more detailed presentation is in [Hong, Mozetic & Michalski 86].

All these systems are based on the AQ algorithm, which generates decision rules from a set of examples, as originally described in [Michalski 69] and [Michalski & McCormick 71]. When building a decision rule, AQ performs a heuristic search through a space of logical expressions to determine those that account for all positive examples and no negative examples. Because there are usually many such *complete* and *consistent* expressions, the goal of AQ is to find the most preferred one, according to a flexible extra-logical criterion. This criterion is defined by the user to reflect the needs of the application domain. When input data may include inconsistent and/or incorrect learning events, it may be advantageous to develop *incomplete* and/or *inconsistent* descriptions. We tested this hypothesis using the TRUNC method of rule reduction and obtained results that were quite unexpected. The results seem to indicate that the TRUNC method may be useful not only for learning from inconsistent and incorrect examples, but also for learning from perfect examples. The method is described in sections 3 and 4, and the results in section 5.

Learning examples are given in the form of *events*, which are vectors of attribute values. Attributes may be of three types: nominal, linear or structured (the domain is a hierarchy). Events represent different decision classes or, generally, concepts. Events from a given class are considered its *positive examples*, and all other events are considered its *negative examples*. For each class a decision rule is produced that covers all positive examples and no negative ones. Rules are represented in VL_1 (Variable-valued Logic system 1) notation [Michalski & Larson 75]. VL_1 is a multiple-valued logic attributional calculus with typed variables. A *selector* relates a variable to a value or a disjunction of values, e.g.:

[Weather_type = cloudy $\vee$ rain]

A conjunction of selectors forms a *complex*. The following complex states that the weather is cloudy, the temperature is greater than 60 degrees, and winds blow from the South or West:

[Weather_type = cloudy] & [Temp > 60] & [Wind_direction = South $\vee$ West]

Complexes are assembled into *covers*. A cover is a disjunction of complexes describing all positive examples and none of the negative examples of the concept. A cover is formed for each decision class separately. It defines the condition part of a corresponding decision rule. The following are two examples of decision rules:

[Transport = car] $\Leftarrow$ [Weather_type = cloudy $\vee$ rain] $\vee$ [Temp = 40..60]

[Transport = bike] $\Leftarrow$ [Weather_type = sun] & [Temp > 60]

As one can see, the rules are easy to interpret. This ease of interpreting AQ15 generated rules is one of the most attractive features of the program. The major idea behind the covering algorithm is to generate a cover in steps, each step producing one conjunctive term (complex) of the cover. Each step starts with focusing attention on one selected positive example (a *seed*). The algorithm generates a set of all complexes (a *star*) which cover the seed and do not cover any negative examples, and then selects the best complex from the star according to the user defined criteria. The basic *covering algorithm* is as follows:

```
While  partial cover does not cover all positive examples
do     1. select an uncovered positive example (a seed),
       2. determine maximally general complexes covering the seed and no negative
          examples (generate a star),
       3. select the best complex from the star according to the user-defined
          problem-dependent preference criteria,
       4. generate a new partial cover by adding the best complex to the current cover.
At the end, a partial cover becomes a cover of the class.
```

The algorithm starts with an initial cover that is either empty, was previously learned, or is supplied by the user. Extending the seed against all the negative examples, i.e. *generating a star* in step 2, is again a multistep procedure which can be described as follows:

```
While  partial star covers some negative examples
do     1. select a covered negative example,
       2. generate all maximally general hypotheses that cover the seed and exclude
          the negative example; the resulting set is called a partial star of the seed
          against the negative example,
       3. generate a new partial star by intersecting the current partial star
          with the partial star of the seed against the negative example,
       4. trim the partial star if the number of disjoint complexes exceeds
          the user defined threshold (the maxstar parameter).
At the end, a partial star becomes the star of the seed, i.e., the set of maximally
general complexes covering the seed and not covering any negative example.
```

The procedure starts with an initial star which is either the entire event space or a complex from the initial cover. If the star generating procedure were to work exhaustively, the search space for covers might grow very rapidly with the number of negative examples and the number of variables used. To deal with this problem, a parameter (*maxstar*) controls how many disjoint complexes may be kept in a partial star. If the number of its disjoint complexes exceeds the parameter, the star is trimmed according to the user specified *criteria*. A typical criterion is: first "maximize the number of positive examples covered" and then, in the case of a tie, "minimize the number of selectors" or "minimize the total cost of variables used".

The program is able to produce rules of different degrees of *generality*. Rules may be *general* (having minimum number of variables, each with maximum number of disjunctive values), *minimal* (minimum number of both, variables and values), or *specific* (maximum number of variables, each with minimum number of values).

AQ15 has the *incremental learning* facility. The user may supply his decision hypotheses as initial rules. The system implements the method of *learning with full memory*. In this type of learning the system remembers all learning examples that were seen so far, as well as the rules it formed. By this method, as opposed to *learning with partial memory*, new decision rules are guaranteed to be correct with respect to all (old and new) learning examples [Reinke 84, Reinke & Michalski 86].

When learning from inconsistent examples, the system provides three options: a) inconsistent examples are treated as positive examples, b) as negative examples, or c) are removed from the data; in this case their membership is decided by the learning process. If statistical information about the probability of inconsistent examples is available, they are preclassified according to the maximum likelihood [Michalski & McCormick 71].

A form of *constructive induction* is implemented in AQ15 as well. The program's background knowledge is expressed in the form of rules, used to generate new attributes not present in input data. The background knowledge rules are of two types: L-rules (logic) that define values of new variables by logical expressions, and A-rules (arithmetic) that introduce new variables as arithmetic functions of original variables. The L-rules and A-rules are two different representations of domain knowledge relevant to the learning process. The L-rules permit one to represent background concept definitions, constraints among the concepts, concept generalization hierarchies, causal dependencies, etc. Concepts known to the program or learned by the program are also added to the stock of L-rules. The algorithm attempts to use new variables to produce better decision rules. The following is an example of a simple L-rule:

$$[\text{Temp} < 32] \implies [\text{Weather\_type} \neq \text{rain}] \; \& \; [\text{Amount\_of\_rain} = \text{NA}]$$

The program is also capable of automatically testing the learned rules on new events. It produces a *confusion matrix* that shows for each concept and event the degree of match, according to the flexible rule interpretation method (see Section 4). Thus it seems to be an ideal tool for experimenting on inductive knowledge acquisition in a variety of practical domains. AQ15 is implemented in Berkley Pascal and runs under the Unix operating system on VAX and SUN machines. It consists of approximately 13.000 lines of code.

3. TRUNCATION OF RULES AND FLEXIBLE MATCHING

Most human concepts are structures with flexible, imprecise boundaries. They can match different instances with varying degrees of precision and have context-dependent meaning. Flexible boundaries permit one to use concepts beyond the typical range; imprecise boundaries are useful for avoiding superfluous or undesirable precision. When building a learning or inference system, two crucial issues are the way in which concepts are represented, and the way in which they are recognized.

As pointed out in [Michalski 86], the meaning of a concept can be distributed between its *base representation* and the *method of its interpretation*. The base representation explicitly states the typical, context-independent properties of the concept. The interpretation method determines whether a given instance satisfies the base concept description by conducting inference - deductive, analogical or inductive - using contextual information and background knowledge. The method may give a yes-no answer or may determine the degree to which the instance satisfies the base concept representation.

Such a *two-tiered* concept representation yields a spectrum of possibilities. At one extreme, all the concept properties are explicitly defined, including any concept variations and exceptions.

This may lead to a very complex and unwieldy concept representation. The concept recognition process, however, would involve merely a simple matching of the properties of an instance with the information in the concept description. At the other end of the spectrum, the concept is explicitly represented only by a simple prototypical description characterizing its ideal form. Such a prototypical description does not have to relate to a single object like in the family resemblance case [Rosch & Mervis 75, Murphy & Medin 85] but may be an abstract concept specification (e.g., a logical formula involving disjunction). The process of concept recognition using a prototypical description is more complicated. Instead of seeking a strict match (a satisfaction of a complex description), the system determines the degree of similarity between the prototypical (ideal) concept description and the given instance, and compares it with the results from matching the instance with other ideal concept descriptions. The concept that gives the best match is assigned to the instance. This method saves memory for concept representation at the expense of more complicated matching procedure. The matching procedure may be the same for a class of concepts, which increases the cost-effectiveness. Also, by changing the concept interpretation method one may affect the concept recognition process *without* changing the concept representation, and thus may apply the concept to new situations, not originally planned.

Depending on the costs associated with storing a representation and performing the inference, the most effective distribution of meaning between the concept representation and interpretation corresponds to some point within the above spectrum. Interesting research problems are to determine this point of optimal balance, and to find out what concept interpretation methods should be used in different situations. Some preliminary experimental results on the last problem are discussed in [Michalski & Chilausky 80], and more recently in [Uhrik 85].

Let us illustrate the above ideas by the knowledge representation used in AQ15. In this program concepts are represented by a disjunction of conjunctive expressions (complexes). Each expression is associated with a pair of weights: t and u, representing the *total* number of instances (events) explained by the expression, and the number of events explained *uniquely* by that expression, respectively. The complexes are ordered according to decreasing values of the t-weight. The t-weight may be interpreted as a measure of the typicality or the representativeness of a complex as a concept description. The complex with the highest weight (t-weight) may be interpreted as describing the most typical examples of the concept. It may also be viewed as a prototypical or the ideal definition of the concept. On the other hand the complexes with lowest u-weight can be viewed as describing rare, exceptional cases. If the learning events from which rules are derived are noisy, such "light" complexes may be indicative of errors in the data.

Two methods of recognizing the concept membership of an instance are distinguished: the *strict* match and the *flexible* match. In the strict match, one tests whether an instance satisfies condition part of a rule (or, generally, if it can be logically derived from it). In the flexible match, one determines the degree of similarity or conceptual closeness between the instance and the condition part. Using the strict match, one can recognize a concept without checking other candidate concepts, i.e., without taking into consideration the context. In the flexible match, one needs to perform inference involving an event and candidate rules, and determine the most similar concept that best "matches" the instance. The flexible matching can be accomplished in a variety of ways, ranging from approximate matching of features through deduction and analogy.

to *conceptual cohesiveness* that employs inductive inference [Michalski & Stepp 83].

The above weight-ordering of complexes suggests an interesting possibility. Suppose we have a t-weight ordered disjunction of complexes, and we remove from it the lightest complex. So truncated description will not strictly match events that uniquely satisfy the truncated complex. However, by applying a flexible match, these events may still come out to be the most closely related to the correct concept, and thus be correctly recognized. A truncated description is, of course, simpler but carries a potentially higher risk of recognition error, and requires a more sophisticated evaluation. We can proceed further and remove the next "light" complex from the cover, and observe the performance. Each such step produces a different trade-off between the complexity of the description on one side, and the risk factor and the evaluation complexity on the other (Figure 1). At some step the best overall result may be achieved for a given application domain. This method of knowledge reduction by truncating ordered covers and applying a flexible matching is called TRUNC.

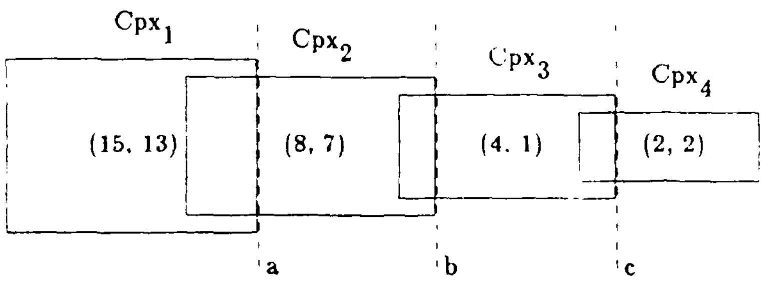


Figure 1. An example of a t-ordered cover. The cuts at a, b and c mark truncated covers with 1, 2 or 3 complexes, respectively. In each pair (x,y), x represents the t-weight, and y represents the u-weight.

The above described trade-off is related to the issues studied in Variable Precision Logic, which is concerned with trade-offs between certainty, computational costs and specificity of inferences [Michalski & Winston 86]. An interesting problem is to test how the cover truncation method affects the accuracy of recognition and the complexity of the decision rules in different practical settings. Section 5 presents results of some such experiments, which in some cases came out very surprising. We now turn to the problem of flexible matching used in this study, and the resolution of a conflict when several concept descriptions are satisfied by an event.

4. FLEXIBLE RULE INTERPRETATION

When strictly matching a new event against a set of (disjunctive) rules, three outcomes are possible: only one rule may be matched (satisfied), more than one rule may be matched, or no rule may match. These cases are classified into categories called SINGLE, MULTIPLE and NO_MATCH, respectively (Figure 2). Each category requires a different evaluation procedure, and a different method of determining the accuracy of concept recognition. For exact match (category SINGLE), the evaluation is easy: the decision is counted as correct if it is equal to the known diagnosis of the testing object, and as wrong otherwise. If there are several exact matches (the MULTIPLE case) or none (the NO_MATCH case) the system activates the *approximate, context-dependent scheme* that determines the best decision (or the most probable one). Comparing this decision with the decision provided by experts, one evaluates it as correct or incorrect. The scheme consists of two simple heuristic evaluation criteria, one for the MULTIPLE case, and the other for the NO_MATCH case.

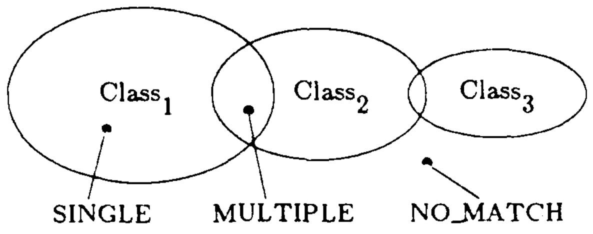


Figure 2. The three possible cases when matching a new event against a set of decision rules.

Estimate of probability for the MULTIPLE case (EP). When an event matches a few rules the system selects the one which suggests the most probable decision. Let $C_1, \ldots, C_n$ denote decision classes and e an event to be classified. For each decision class C_i we have a rule that consists of a disjunction of complexes (*Cpx*), which, in turn are conjunctions of selectors (*Sel*). We define the estimate of probability, *EP*, as follows:

1) *EP* of a complex Cpx_j in the context of the event e is the ratio of the weight of the complex (the number of positive learning examples covered by the complex) by the total number of learning examples (*#examples*), if the complex is satisfied by the event e, and equals 0 otherwise:

$$EP(Cpx_j, e) = \begin{cases} Weight(Cpx_j)\ /\ \#examples & \textit{if complex } Cpx_j \textit{ is satisfied by } e, \\ 0 & \textit{otherwise.} \end{cases}$$

2) *EP* of a class C_i is the probabilistic sum of *EP*s of its complexes. If the rule for C_i consists of a disjunction of two complexes $Cpx_1 \vee Cpx_2$, we have:

$$EP(C_i, e) = EP(Cpx_1, e) + EP(Cpx_2, e) - EP(Cpx_1, e) \times EP(Cpx_2, e)$$

The most probable class is the one with the largest *EP*, i.e., the one whose satisfied complexes cover the largest number of learning examples. It is assumed that the learning examples are a representative sample of the domain, and that the numbers of examples for each class are proportional to the frequency of occurence of classes. Obviously, if the class is not satisfied by the given event, its *EP* equals 0. For each C_i this measure determines the number of learning examples that support the classification of the new event into class C_i. The larger such number is, the stronger support is assumed.

Measure of fit for the NO_MATCH case (MF). In this case the event belongs to a part of the event space that is not covered by any decision rule and this calls for flexible matching. One way to perform such matching is to measure the fit between attribute values in the event and the class description, taking into consideration the prior probability of the class. We used in the experiments a simple measure, called *measure of fit, MF*, defined as follows:

1) MF of a selector Sel_k and an event e is 1, if the selector is satisfied, i.e. if one of event's attribute values lies in the range of values of the selector. Otherwise, this measure is proportional to the amount of the decision space covered by the selector:

$$MF(Sel_k, e) = \begin{cases} 1 & \textit{if selector } Sel_k \textit{ is satisfied by } e, \\ \dfrac{\#Values}{DomainSize} & \textit{otherwise.} \end{cases}$$

where $\#Values$ is the number of disjunctively linked attribute values in the selector, and $DomainSize$ is the total number of the attribute's possible values.

2) MF of a complex Cpx_j to an event e is defined as the product of MFs for a conjunction of its constituent selectors, weighted by the proportion of learning examples covered by the complex:

$$MF(Cpx_j, e) = \prod_k MF(Sel_k, e) \times (Weight(Cpx_j) \; / \; \#examples)$$

3) MF of a class C_i to en event e is obtained as a probabilistic sum for a disjunction of complexes. If the rule for C_i consists of a disjunction of two complexes $Cpx_1 \vee Cpx_2$, we have:

$$MF(C_i, e) = MF(Cpx_1, e) + MF(Cpx_2, e) - MF(Cpx_1, e) \times MF(Cpx_2, e)$$

We can interpret the measure of best fit of a class as a combination of "closeness" of the event to the class and an estimate of the prior probability of the class. Closeness is measured by MF of selectors, where the fit is complete for selectors that are satisfied. MF of an unsatisfied selector is the probability that it will be satisfied if the event's corresponding attribute value changes. A selector that covers more decision space fits an event better than a selector that covers less decision space (having fewer alternative values). Closeness to a complex is the probability that the event will be covered by the complex if the values of attributes corresponding to unsatisfied selectors change. MF of a complex is then weighted by an estimate of priori probability, i.e., the proportion of the learning examples that it covers. Note that the estimate of probability EP is a special case of the measure of fit MF; when all selectors in a complex are satisfied the measure of fit of a complex is the same as the estimate of probability.

The above measure of fit is one of many possible measures that can be devised for flexible matching. One way to improve this measure would be to define a distance between an attribute value and a selector, when attributes are linear [Michalski & Chilausky 80].

5. EXPERIMENTS

The experiments were performed on data from three medical domains: lymphography, prognosis of breast cancer recurrence and location of primary tumor. All data were obtained from the Institute of Oncology of the University Medical Center in Ljubljana, Yugoslavia [Kononenko, Bratko & Roskar 84].

Lymphography. This domain is characterized by 18 attributes and 4 diagnostic classes. Data of 148 patients were available. The set of attributes was *complete*, i.e., was sufficient for having all learning examples consistent. This means that examples of any two classes were always different. Diagnoses in this domain were not verified and actual testing of physicians was not done. A specialist's estimation is that internists diagnose correctly in about 60% and specialists in about 85% of cases.

Prognosis of Breast Cancer Recurrence. For about 30% of patients that undergo a breast cancer operation, the illness reappears in five years. Prognosis of this recurrence is very important for patients' post-operational treatment. The domain is characterized by 2 decision classes and 9 attributes. The set of attributes was incomplete, i.e., not always sufficient to distinguish between cases with different prognosis. Data for 286 patients with known diagnostic status 5 years after the operation were available. Five specialists of the Institute of Oncology were tested. They gave a correct prognosis in 64% of cases.

Location of Primary Tumor. Physicians distinguish among 22 possible locations of primary tumor. Patients' diagnostic data were described by 17 attributes. The given set of attributes was incomplete, as some patients with the same values of all attributes had different location of primary tumor. Data of 339 patients with known locations of primary tumor (verified by operation or by X-ray) were available for the experiment. At the Institute of Oncology 4 internists and 4 specialists were tested. Internists determined a correct location of primary tumor in 32% and oncologists in 42% of test cases. Regarding these relatively low results, we should stress that there are 22 possible locations and that the correct location of primary tumor is only one of the sources of evidence used in cancer treatment.

Table 1 provides a summary of these medical domains. It presents the number of examples, of classes, of attributes, and the average number of values per attribute for each domain.

In all medical domains 70% of examples were selected for learning and the remaining 30% for testing. Each testing experiment was repeated 4 times with randomly chosen learning examples. Final results are the average of 4 experiments.

Domain	Examples	Classes	Attributes	Values/Attr
Lymphography	148	4	18	3.3
Breast cancer	286	2	9	5.8
Primary tumor	339	22	17	2.2

Table 1. Characteristics of the data for the three medical domains.

For illustration, in Figure 3 there is an example of a paraphrased rule from the domain of lymphography.

```
Diagnosis = lymphoma    if:
   Filling_defects_lacunar = none ∨ lacunar ∨ lacunar_central
   Special_structures_and_forms = none ∨ bladder                        Base
   Lymph_nodes_size_diminishing = 0                                     complex
   Lack_of_lymph_nodes_filling = yes
   No_of_diseased_lymph_nodes ≥ 10                      (t-weight:40, u-weight:22)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
               ∨
   Filling_of_lymp_nodes = grains ∨ fine_drops ∨ dispersed ∨ obscure
   Special_structures_and_forms = cup ∨ bladder
   Early_filling_of_lymp_nodes = yes
   Block_of_afferrent_vessels = no
   By_pass = no                                         (t-weight:24, u-weight:7)
               ∨
   Special_structures_and_forms = cup ∨ bladder
   Lymph_vessels = curves ∨ deformities
   Lymph_nodes_size_enlarged = 1..2
   Block_of_afferent_vessels = no
   Dislocation_of_lymph_nodes = yes                     (t-weight:18, u-weight:3)
               ∨
   Filling_of_lymp_nodes = fine_drops ∨ stripes ∨ obscure
   Filling_defects_various = follicular ∨ gross_central
   Lymph_nodes_size_enlarged = 1..3
   Block_of_lymph_nodes_chain = no
   Extravasates = yes                                   (t-weight:10, u-weight:3)
               ∨
   Changes_of_lymph_nodes_shape = oval
   No_of_diseased_lymph_nodes = 30..39                  (t-weight:2, u-weight:1)
```

Figure 3. A complete rule, generated by AQ15 from all available examples, with t-ordered complexes, for the domain of lymphography. The rule consists of 5 complexes and 22 selectors. After truncation to the "base complex" (with the highest t-weight) the rule has only 1 complex with 5 selectors. T-weight is the total number of examples covered by a complex, and u-weight is the number of examples covered by the complex uniquely.

Two sets of experiments were performed. In the first one only rules of the minimal type were used. Different cover reduction mechanisms were applied on them, and their effect on complexity and classification accuracy of rules was determined. Complexity was measured by the total number of selectors and complexes in the rules, and accuracy by the "1st choice correct" evaluation method (Table 2). In the second set of experiments we measured classification accuracy by two parameters: correctness and precision. We used rules of different degree of generality, applied different evaluation methods and used two cover reduction mechanisms to find the optimal combination of correctness and precision (Table 3).

Domain	Cover truncation	Complexity Sel	Complexity Cpx	Accuracy 1st choice	Human Experts	Random Choice
Lymphography	no	**37**	12	**81%**	85% (estimate)	25%
	unique >1	34	10	80%		
	base cpx	**10**	**4**	**82%**		
Breast cancer	no	**160**	**41**	**66%**	64%	50%
	unique >1	128	32	66%		
	base cpx	**7**	**2**	**68%**		
Primary tumor	no	551	104	39%	42%	5%
	unique >1	257	42	41%		
	base cpx	112	20	29%		

Table 2. Average complexity and accuracy of AQ15's rules (minimal type) learned from 70% of examples, over 4 experiments. Two simple cover truncation mechanisms were applied - keeping only complexes that uniquely cover more than one example (unique >1), and deleting all but the heaviest complex in each rule (base cpx).

In addition to results obtained from using complete (untruncated) rules, results of two other experiments are presented. In the first experiment we eliminated from rules all complexes that cover uniquely only one learning example, and in the second we eliminated all complexes except the most representative one that covers the largest number of learning examples. Complexity of rules is measured by the number of selectors and complexes. Table 2 shows that some results came out very surprising. When the cover of each class was truncated to only one (the heaviest) complex, the complexity of the rule set for lymphography went down from the total of 12 complexes and 37 selectors to only 4 complexes (one per class) and 10 selectors (see bold numbers). At the same time the performance of rules went slightly up (from 81% to 82%) ! A similar phenomenon occurred in the breast cancer domain, where the number of selectors and complexes went down from 160 and 41 to 7 and 2, respectively; while the performance went slightly up from 66% to 68%. This means that by using the TRUNC method one may significantly reduce the knowledge base without affecting its performance accuracy. Results for human experts were the average of testing of five and four domain specialists in the domains of breast cancer recurrence and primary tumor, respectively [Kononenko, Bratko & Roskar 84]. In the domain of lymphography, physicians' accuracy is given only as their own estimate; it was not independently measured.

In practice, giving always exactly one answer (1st choice) is often not the most appropriate. One might wish to get more than just one possible diagnosis, or none if there is not enough evidence. If any of the alternative diagnoses given by the system is the same as the known diagnosis of the testing example the answer is counted as a correct one. However, the more alternative diagnoses, the smaller diagnostic precision of the system. Therefore, in evaluation of such a system, the results should be measured by two quantities: *correctness* (the ratio of the number of correct answers by the number of testing examples), and *precision* (the ratio of the number of correct answers by the total number of answers given).

	Evaluation method for		Type of	All cpx		Best cpx	
Domain	MULTIPLE	NO-MATCH	rules	Corr.	Prec.	Corr.	Prec.
Lymphography	1st choice correct	1st choice correct	specific	79%	79%	80%	80%
			minimal	81%	81%	82%	82%
			general	81%	81%	81%	81%
	correct if match	always incorrect	specific	63%	85%	52%	94%
			minimal	78%	77%	58%	89%
			general	86%	74%	58%	87%
	correct if match	1st choice correct	specific	80%	78%	81%	81%
			minimal	83%	76%	83%	82%
			general	89%	74%	82%	81%
Breast cancer	1st choice correct	1st choice correct	specific	68%	68%	67%	67%
			minimal	66%	66%	68%	68%
			general	65%	65%	65%	65%
	correct if match	always incorrect	specific	59%	64%	13%	67%
			minimal	77%	57%	16%	67%
			general	86%	54%	17%	64%
	correct if match	1st choice correct	specific	72%	62%	68%	68%
			minimal	80%	58%	68%	68%
			general	86%	54%	66%	66%
Primary tumor	1st choice correct	1st choice correct	specific	41%	41%	33%	33%
			minimal	39%	39%	29%	29%
			general	39%	39%	29%	29%
	correct if match	always incorrect	specific	33%	31%	22%	44%
			minimal	50%	24%	25%	34%
			general	51%	24%	25%	34%
	correct if match	1st choice correct	specific	47%	34%	35%	33%
			minimal	52%	24%	32%	28%
			general	53%	24%	32%	28%

Table 3. Trade-offs between correctness and precision of AQ15's rules for different evaluation methods and different types of rules.

Several experiments with AQ15 were performed to evaluate trade-offs between correctness and precision (Table 3). This trade-off is a reflection of the phenomena studied in Variable Precision Logic [Michalski & Winston 84]. We tried three different evaluation schemes (representing combinations of 1st choice correct, correct if match and incorrect for MULTIPLE and NO_MATCH cases). The "1st choice correct" means that the best flexible matching was used. The "correct if match" for MULTIPLE and "incorrect" for NO_MATCH mean that the strict match method was used. We also used rules of different degree of generality (specific, minimal, general) and different cover reduction mechanism. "All cpx" and "Best cpx" mean complete cover and the cover truncated to one complex, respectively. One of the interesting future research tasks is to find an appropriate information-theoretic measure for defining an optimal combination of correctness and precision.

6. ANALYSIS OF RESULTS

The domain of lymphography seems to have some strong patterns and the set of attributes is known to be complete, i.e., no event description belongs to more than one class. There are four possible diagnoses, but only two of them are prevailing, i.e., they occur much more often than others. The domain of breast cancer has only two decision classes, but does not have many strong patterns. Domain of location of primary tumor has many decision classes and mostly binary attributes. There are only a few examples per class, and the domain seems to be without any strong patterns. Both domains are underspecified in the sense that the set of available attributes is incomplete (not sufficient to discriminate between different classes). The statistics in Table 4 include average number of complexes per rule, average number of attributes per complex, average number of values per attribute and finally, average number of learning examples covered by one complex. We can see that in the domain of primary tumor decision rules consist of complexes that in average cover slightly more than 2 examples. In the domain of lymphography complexes in average cover 8 examples, which indicates a presence of strong patterns.

Domain	Cpx/Rule	Attr/Cpx	Values/Attr	Examples/Cpx
Lymphography	3	3.1	1.8	8
Breast cancer	20	3.9	1.7	5
Primary tumor	5.2	5.3	1.0	2.3

Table 4. Average complexity of AQ15's decision rules (minimal type) in the three medical domains, when no cover truncation mechanism was applied.

Several experiments with AQ15 were performed, each with a different complex truncation heuristic. This was done in order to investigate the trade-off between complexity and accuracy, and to derive some preliminary conclusions about the effects of the cover reduction mechanism. Results given in Table 2 present only two extreme cases of these experiments. By eliminating all complexes but one, a significant reduction of complexity was obtained. Except for the primary tumor domain, there was no decrease of accuracy.

In the domain of primary tumor, initial elimination of lightest complexes (those that cover only 1 example) increased accuracy from 33% to 41%; accuracy decreased when further complexes were eliminated. In the domain of lymphography accuracy increased until only one "heaviest" complex in the two most important rules was kept (82%). In the breast cancer domain each step of elimination of complexes increased accuracy as well. Best results were obtained when all complexes for the class "recurrence" were deleted. The obtained diagnostic accuracy was 72% which is close to a priori probability of the diagnosis "no recurrence".

It is surprising that a cover reduction mechanism that strongly simplifies the rule base may have no affect on classification accuracy. Removing complexes from a cover is equivalent to removing disjunctively linked conditions from a concept description. This process overspecializes a knowledge representation, producing an *incomplete* concept description (i.e., a one that does not cover some positive examples).

Such knowledge reduction technique by specialization may be contrasted with knowledge reduction by generalization used in the ASSISTANT learning program, a descendant of ID3 [Quinlan 83]. This program represents knowledge in the form of decision trees, and has been applied to the same medical problems as here [Kononenko, Bratko & Roskar 84]. The program applies a *tree pruning* technique based on the principle of maximal classification accuracy. The technique removes certain nodes from a tree, and is equivalent to removing conjunctively linked conditions from a concept description. Thus, such a knowledge reduction technique overgeneralizes the knowledge representation, producing an *inconsistent* concept description (i.e., a one that covers some negative examples). It is interesting to point out that this technique may also lead to an improvement of accuracy in decision making when learning from noisy and overlapping data. Table 5 presents the complexity and diagnostic accuracy of ASSISTANT's trees built with and without tree pruning [Kononenko, Bratko & Roskar 84]. Complexity of trees is given by the number of nodes and leaves. In all domains results were better when the tree pruning mechanism was used.

	Tree	Complexity		Accuracy
Domain	pruning	Nodes	Leaves	1st choice
	no	38	22	76%
Lymphography	yes	25	14	77%
	no	120	63	67%
Breast cancer	yes	16	9	72%
	no	188	90	41%
Primary tumor	yes	35	18	46%

Table 5. Average complexity and accuracy of decision trees built by ASSISTANT on 70% of examples, over 4 experiments. In all three domains the tree pruning mechanism reduced the complexity and increased the accuracy. Note that more than one decision may be assigned to some leaves (hence there are only 18 leaves for 22 classes in the primary tumor case).

Tree pruning corresponds to the removal of selectors from complexes. This seems to suggest that when learning from noisy or inconsistent examples the knowledge reduction process may not only involve removal of complexes from a cover (a specialization process) but also removal of selectors from complexes (a generalization process). This means that the generated concept description would be both inconsistent and incomplete. It is an interesting problem for further research to determine conditions under which such inconsistent and incomplete descriptions might be more advantageous than consistent and complete ones.

7. CONCLUSION

A major contribution of the paper is a demonstration that a relatively simple, attribute-based inductive learning method is able to produce decision rules of sufficiently high quality to be applicable to practical problems with noisy, inconsistent and/or incompletely specified learning examples. An especially important for practical applications is perhaps the fact that the method produced these results without using a large amount of domain knowledge that would be required by an analytic approach or explanation-based generalization [Mitchell, Keller & Kedar-Cabelli 86; DeJong & Mooney 86]. It relied primarily on learning examples that were obtained from already existing records or human experts. It is well known that it is typically easier for an expert to make decisions (i.e., to produce examples) than to formulate a theory justifying them.

Although the program can work with relatively little domain knowledge (e.g., only the specification of types and domains of attributes, and the preference criterion), it can also take advantage of the domain knowledge when it is available. The latter is realized by employing background knowledge representation facilities in the form of logical and arithmetical rules (L rules and A-rules).

The AQ15 program has shown itself to be a powerful and flexible tool for experimenting with inductive knowledge acquisition. It produces decision rules which are easy to interpret and comprehend. The knowledge representation in the program is, however, limited to only attributional descriptions. For problems that require structural descriptions one may use a related program INDUCE2 [Hoff, Michalski & Stepp 83] or its incremental learning version INDUCE4 [Mehler, Bentrup & Riedsel 86]. A weakness of the experimental part of the paper is that the authors had no influence on the way the data were prepared for the experiments and the available data allowed us to test only a few of the features of AQ15.

Another major result is a demonstration that the knowledge reduction by truncating the covers may lead in some cases to a substantial reduction of the rule base without decreasing its performance accuracy. We have also shown that by varying the degree of generality of rules and applying different evaluation methods, different trade-offs between the correctness and precision of decision rules are achieved. Further research will be required to find for any given domain a rule reduction criterion that leads to the best trade-off between accuracy and complexity of a rule base. Another topic for further research is to develop more sophisticated methods for flexible matching.

ACKNOWLEDGEMENTS

The authors thank Ivan Bratko and Igor Kononenko from the Faculty of Electrical Engineering at the E. Kardelj University in Ljubljana for collaboration and comments, and physicians Matjaz Zwitter and Milan Soklic from the Institute of Oncology at the University Medical Center in Ljubljana for providing medical data and helping to interpret them. We further acknowledge Gail Thornburg from the UI School of Library and Information Science and the AI Laboratory at the Dept. of Computer Science for her criticism and valuable suggestions.

REFERENCES

DeJong G., Mooney, R., Explanation-Based Learning: An Alternative View. *Machine Learning*, Vol. 1, No. 2, 1986.

Hoff, W., Michalski, R.S., Stepp, R.E., INDUCE.2: A Program for Learning Structural Descriptions from Examples. Report ISG 83-4, UIUCDCS-F-83-904, Dept. of Computer Science, University of Illinois at Urbana-Champaign, 1983.

Hong, J., Mozetic, I., Michalski, R.S., AQ15: Incremental Learning of Attribute-Based Descriptions from Examples, the Method and User's Guide. Report ISG 86-5, UIUCDCS-F-86-949, Dept. of Computer Science, University of Illinois at Urbana-Champaign, 1986.

Kononenko, I., Bratko, I., Roskar, E., Experiments in Automatic Learning of Medical Diagnostic Rules. Presented at the *International School for the Synthesis of Expert Knowledge Workshop '84*, Bled, Yugoslavia, August 22-24, 1984. Also Technical Report, Faculty of Electrical Engineering, E. Kardelj University, Ljubljana, Yugoslavia, 1984.

Mehler, G., Bentrup, J., Riedesel J., INDUCE.4: A Program for Incrementally Learning Structural Descriptions from Examples. Report in preparation, Dept. of Computer Science, University of Illinois at Urbana-Champaign, 1986.

Michalski, R.S., On the Quasi-Minimal Solution of the General Covering Problem. *Proceedings of the V International Symposium on Information Processing (FCIP 69)*, Vol. A3 (Switching Circuits), Bled, Yugoslavia, pp. 125-128, 1969.

Michalski, R.S., Theory and Methodology of Machine Learning. In R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds.), *Machine Learning - An Artificial Intelligence Approach*. Palo Alto: Tioga, 1983.

Michalski, R.S., Two-tiered Concept Representation, Inferential Matching and Conceptual Cohesiveness. An invited paper for the *Workshop on Similarity and Analogy*, Allerton House, University of Illinois, June 12-14, 1986.

Michalski, R.S., Chilausky, R.L., Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis. *International Journal of Policy Analysis and Information Systems*, Vol. 4, No. 2, pp. 125-161, 1980.

Michalski, R.S., Larson, J., AQVAL/1 (AQ7) User's Guide and Program Description. Report No. 731, Dept. of Computer Science, University of Illinois at Urbana-Champaign, 1975.

Michalski, R.S., McCormick, B.H., Interval Generalization of Switching Theory. Report No. 442, Dept. of Computer Science, University of Illinois at Urbana-Champaign, 1971.

Michalski, R.S., Stepp, R.E., Learning from Observations: Conceptual Clustering. In R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds.), *Machine Learning - An Artificial Intelligence Approach*. Palo Alto: Tioga, 1983.

Michalski, R.S., Winston, P.H., Variable Precision Logic. Artificial Intelligence Memo No. 857, MIT, Cambridge. An extended version to appear in *AI Journal*, 1986.

Mitchell, T.M., Keller, R.M., Kedar-Cabelli, S.T., Explanation-Based Generalization: A Unifying View. *Machine Learning*, Vol. 1, No. 1, pp. 47-80, 1986.

Mozetic, I., Knowledge Extraction through Learning from Examples. In T.M. Mitchell, J.G. Carbonell, R.S. Michalski (Eds.), *Machine Learning: A Guide to Current Research*. Kluwer Academic Publishers, 1986.

Murphy, G.L., Medin, D.L., The Role of Theories in Conceptual Coherence. *Psychological Review*, Vol. 92, No. 3, 1985.

Quinlan, J.R., Learning Efficient Classification Procedures and their Application to Chess End Games. In R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds.), *Machine Learning - An Artificial Intelligence Approach*. Palo Alto: Tioga, 1983.

Reinke, R.E., Knowledge Acquisition and Refinement Tools for the ADVISE META-EXPERT System. M.S. Thesis, ISG 84-4, UIUCDCS-F-84-921, Dept. of Computer Science, University of Illinois at Urbana-Champaign, 1984.

Reinke, R.E., Michalski, R.S., Incremental Learning of Decision Rules: A Method and Experimental Results. To appear in *Machine Intelligence 11* (eds. J.E. Hayes, D. Michie, J. Richards), Oxford University Press, 1986.

Rosch, E., Mervis, C.B., Family Resemblance: Studies in the Internal Structure of Categories. *Cognitive Psychology*, No. 7, 1975.

Uhrik, T.K., A Rule Exerciser for Knowledge Base Enhancement in Expert Systems. M.S. Thesis, Dept. of Computer Science, University of Illinois at Urbana-Champaign, 1985.

FILM